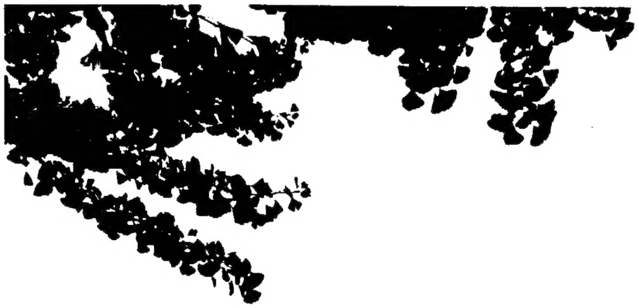
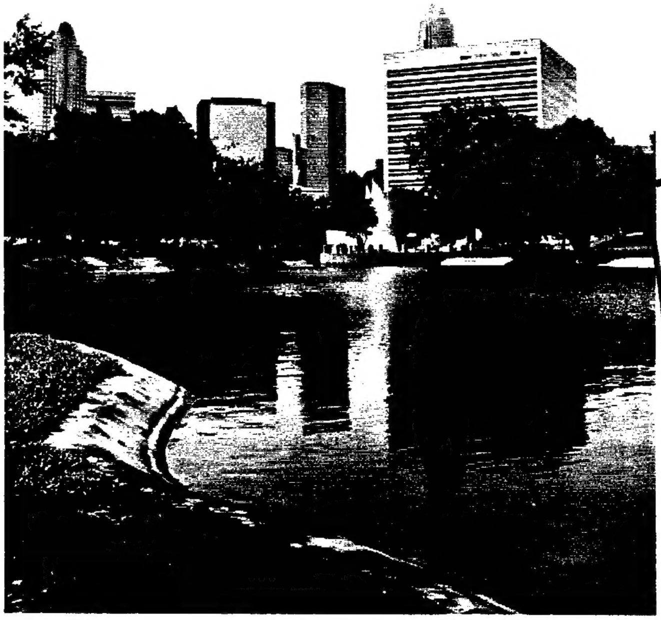
ISMIS-94





Proceedings of the Eighth International Symposium on Methodologies for Intelligent Systems

October 16–19, 1994

Charlotte, North Carolina

This work relates to Department of Navy Grant N00014-94-1-0799 issued by the Office of Naval Research. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

These Proceedings were assembled by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

TABLE OF CONTENTS

POSTER SESSIONS

INTELLIGENT INFORMATION SYSTEMS

"Adaptive Information Filtering by Monitoring User Behavior," Max Höefferer, Bernd Knaus, and Werner Winiwarter (Vienna, Austria)

"Intelligent Query Processing in Object-Oriented Databases," Suk-Chung Yoon (Widener University) and Il-Yeol Song (Drexel University)

"Agent-Oriented Information System Design," Egon M. Verharen and Hans Weigand (Tilburg University, The Netherlands)

"A Linguistic Geometry for Intelligent Systems," Boris Stilman (University of Colorado, Denver)

KNOWLEDGE REPRESENTATION AND INTEGRATION

"Reasoning from Data in the Mathematical Theory of Evidence," Meiczysław A. Kłopotek (ICS PAS Warsaw, Poland)

"Effects of Attribute Replication in Inheritance Systems: Some Empirical/Simulation Results," Aboalfazl Salimi, Fernando Gomez, and Ali Orooji (University of Central Florida)

"A Knowledge-Based Scene Analysis System for Aerial Image," Crystal J. Su (IBM Canada) and Zhongquan Wu (ESTEK Corporation, North Carolina)

"Integration of Rule Inferences Into a Terminological Component," Peter Forster and Bernd Novotny (University of Stuttgart, Germany)

"Reasoning About Numbers in Tecton," Deepak Kapur and Xumin Nie (State University of New York at Albany)

LOGIC FOR ARTIFICIAL INTELLIGENCE

"A Transformation Technique to Combine the Linear and the Unit-Resulting Restrictions," Peter Baumgartner (University of Koblenz, Germany)

"A Note of Tableaux of Logic of Paradox," Zuoquan Lin and Wei Li (Shantou University, China)

METHODOLOGIES/APPLICATIONS

"Knowledge Modelling for Evolutionary Program Synthesis," Jutta Eusterbrock GMD Darmstadt, Germany)

"Process Fault Diagnosis for a Magnetic Levitation Control System Based on an Artificial Neural Network," Ching-Yu Tyan, Paul P. Wang (Duke University), and Dennis R. Bahler (North Carolina State University)

Preface

This volume contains papers which were selected for presentation at the Poster Session of the Eight International Symposium on Methodologies for Intelligent Systems - ISMIS'94, held in Charlotte, North Carolina, October 16-19, 1994. The symposium was hosted by the University of North Carolina at Charlotte and sponsored by Office of Naval Research, Oak Ridge National Laboratory, MITRE Corporation, UNC-Charlotte and IBM-Charlotte.

ISMIS is a conference series that was started in 1986 in Knoxville, Tennessee. It has since then been held in Charlotte (North Carolina), once in Knoxville, once in Torino (Italy) and once in Trondheim (Norway).

The Program Committee has decided to select the following major areas for ISMIS'94:

- Approximate Reasoning
- Evolutionary Computation
- Intelligent Information Systems
- Knowledge Representation and Integration
- Learning and Adaptive Systems
- Logic for Artificial Intelligence
- Methodologies (modeling, design, validation)

The contributed papers were selected from more than 120 full draft papers by the following Program Committee: Luigia Carluci Aiello (Roma U., Italy), Alan Biermann (Duke U.), Piero Bonissone (GE), Jacques Calmet (U. Karlsruhe, Germany), John Campbell (U. College - London, England), Jaime Carbonell (CMU), Wesley Chu (UCLA), Misbah Deen (Keele U., UK), Robert Demolombe (CERT-Toulouse, France), Eric Dietrich (SUNY Binghamton), Jon Doyle (MIT), Douglas Fisher (Vanderbilt U.), Michael Georgeff (AAII, Australia), David Hislop (US Army Research Office), Matthias Jarke (RWTH Aachen, Germany), Leo Joskowicz (IMB - Yorktown Heights), Yves Kodratoff (U. Paris VI, France), Jan Komorowski (U. Trondheim, Norway), Kurt Konolige (SRI), Catherine Lassez (IBM Yorktown Heights), Michael Lowry (NASA Ames), Robert Meersman (Tilburg U., Netherlands), Zbigniew Michalewicz (UNC-C), Ryszard Michalski (George Mason), Jack Minker (U. Maryland), Masao Mukaidono (Meiji U., Japan), Lin Padgham (Linkoping U., Sweden), Rohit Parikh (CUNY), Zdzislaw Pawlak (Warsaw U. Tech., Poland), Francois Pin (ORNL), Henri Prade (U. Paul Sabatier, France), Luc De Raedt (Catholic U. Leuven, Belgium), Zbigniew Ras (UNC-C), Lorenza Saitta (U. Torino, Italy), Erik Sandewall (Linkoping U., Sweden), Timos Sellis (National Tech. U. Athens, Greece), Yoav Shoham (Stanford U.), Sal Stolfo (Columbia U.), Richmond Thomason (U. Pittsburgh), Ralph Wachter (ONR), Marianne Winslett (U. Illinois - Urbana), Carlo Zaniolo (UCLA), Maria Zemankova (MITRE), Jan Zytkow (Wichita State). Additionally, we acknowledge the help in reviewing the papers from: Yongyuth Aramkulchai (Imperial College, UK), Chitta Baral (U. Texas - El Paso), Christer Baeckstroem (Linkoping U., Sweden), Marianne Baudinet (U. Libre de Bruxelles, Belgium), Pierre Berlandier (INRIA, France), Sanjiv Bhatia (U. Missouri - St. Louis), Isabelle Borne (Open U., England), Patricia Carbone (MITRE), Steve Chenoweth (NCR), Jan Chomicki (Kansas State U.), Bill Chu (UNC-C), Robin Cockett (U. Calgary, Canada), Luca Console (U. Torino, Italy), Ray D'Amore (MITRE), L. Degerstedt (Linkoping U., Sweden), Luis Farinas del Cerro (U. Calabria, Italy), Bipin Desai (Concordia U., Canada), Olga De Troyer (Tilburg U., Netherlands), Rose Dieng (INRIA, France), Dimiter Driankov (Linkoping U., Sweden), Christoph Eick (U. Houston), Peter Flach (Tilburg U., Netherlands), Adam Gadomski (ENEA, Italy), Yolanda Gil (ISI/USC), John Grant (Towson State U.), Jerzy Grzymala-Busse (U. Kansas), Sablon Gunther (Catholic U. Leuven, Belgium), Tibor Gyires (Illinois State U.), Mirsad Hadzikadic (UNC-C), Lucja Iwanska (Wayne State), Cezary Janikow (U. Missouri -St. Louis), Yuejun Jiang (Imperial College, UK), Debby Keen (U. Kentucky), Mieczyslaw Kokar (Northeastern U.), Diana Komnenovic (Keele U., UK), Akhil Kumar (Cornell U.), Sukhamay Kundu (LSU), Guillaume Le Blanc (U. Paris VI, France), Jorge Lobo (U. Illinois - Chicago), Anthony Maida (U. SW Louisiana), Jacek Maitan (UNC-C), M. Matskin (Trondheim U., Norway), T.L. McCluskey (U. Huddersfield, UK), Artur Mikitiuk (U. Kentucky), Neil Murray (SUNY-Albany), Claire Nedellec (U. Paris VI, France), Ulf Nilsson (Linkoping U., Sweden), Madhura Nirkhe (Florida Atlantic U.), E.M. Oblow (ORNL), Levent Orman (Cornell U.), Luigi Palopoli (U. Calabria, Italy), Sergey Petrov (ORNL), Gregory Piatetsky-Shapiro (GTE Lab.), Jan Plaza (U. Miami), Luigi Portinale (U. Torino, Italy), Arcot Rajasekar (U. Kentucky), Satyendra Rana (Wayne State), Helena Rasiowa (U. Warsaw, Poland), Robert Reynolds (Wayne State), Celine Rouveirol (U. Paris VI, France), Pasquale Rullo (U. Calabria, Italy), Steven Salzberg (Johns Hopkins U.), Francesco Scarcello (U. Calabria, Italy), Michele Sebag (Ecole Polytech., France), Shashank Shekhar (ORNL), Andrzej Skowron (U. Warsaw, Poland), Roman Slowinski (Tech. U. Poznan, Poland), Roman Swiniarski (San Diego State U.), Bhavani Thuraisingham (MITRE), Pietro Torasso (U. Torino, Italy), C. Vasudevan (Florida Atlantic U.), Gilles Venturini (U. Paris VI, France), Hans Weingand (Tilburg U., Netherlands), S.K. Michael Wong (U. Regina, Canada), Wlodek Zadrozny (IBM - Yorktown Heights), Wojtek Ziarko (U. Regina, Canada).

The Symposium was organized by the Department of Computer Science, University of North Carolina at Charlotte. The Organizing Committee consisted of Bill Chu, Mirsad Hadzikadic, Karen Harber, Linda Kroff, Rick Lejk, M.S. Narasimha and Zbigniew Ras.

We wish to express our thanks to William Campbell (NASA), Jaime Carbonell (CMU), Keh-Hsun Chen (UNC-C), Ed Fox (Virginia Tech.), B. Chandrasekaran (Ohio State) and Robert Meersman (Tilburg U., Netherlands) who presented the invited addresses at the symposium. We would like to express our appreciation to the sponsors of the symposium and to all who submitted papers for presentation and publication in the proceedings. Special thanks are due to Francois Pin (ORNL) for his help and support.

Finally, we would like to thank Magda Szymanowska (East Carolina Univ.) whose contribution to organizing this symposium was essential to its becoming a success.

Z.W. Ras, M. Zemankova

July 1994

A Transformation Technique to Combine the Linear and the Unit-Resulting Restrictions*

Peter Baumgartner

University of Koblenz, Rheinau 1,
D 56075 Koblenz, Germany
E-mail: peter@informatik.uni-koblenz.de
Phone: +49–261–9119–426

Abstract

We present a method by which a Horn clause set can be transformed into an efficient inference system. This system combines the linearity restriction (as found e.g. in Prologs SLD-resolution) with the unit-resulting restriction (i.e. the resolvent is a unit). It can be used in combination with linear calculi such as e.g. (theory) model elimination to yield a sound, complete and efficient calculus for full first order logic. The transformation itself is a completion procedure in spirit of Knuth-Bendix completion.

1 Introduction

In proving mathematical theorems a problem can often be divided into two parts: the one part, called the *theory*, describes common knowledge about some problem domain (e.g. equality, orderings, set theory, arithmetic). The other part consists of the *hypothesis* and the concrete *theorem* to be proved.

This distinction can be modelled within automated theorem proving by *theory reasoning calculi* such as theory resolution [Sti85] or theory model elimination [Bau92, Bau93b]. There, a "foreground reasoner" (such as model elimination) is coupled with a "background reasoner" for reasoning within the theory. In this scenario, it would be most useful to transform a theory "once and for all" into a background reasoner which can process the theory far more efficiently than it would be possible by supplying the theory axioms as input clauses. Consider for example a transitivity axiom such as $\neg(x < y) \vee \neg(y < z) \vee (x < z)$ for strict orderings. This axiom can be used e.g. in ordinary resolution or model elimination in almost any proof state and leads to an explosion of the search space.

*This research was sponsored by DFG within the "Schwerpunktprogramm *Deduktion*". It is a condensed version of [Bau93a]. That paper also contains all proofs in full length.

It is the purpose of this paper to define a new and general technique, which transforms a given (Horn) theory into an inference system for background reasoning within theory reasoning calculi. Currently, the transformation technique is tailored towards the use with *linear* calculi such as theory model elimination. Our interest in linear calculi comes from their successful application in automated theorem proving (see e.g. [BF93b, LSBB92, AS92] for descriptions of running systems; the calculus of model elimination was introduced by Loveland [Lov68, Lov78], a recent variant was defined in [BF93a]).

In this paper we will concentrate on the transformation technique alone; its place within e.g. theory model elimination is indicated in figure 1. A detailled description of the interface between the two reasoning components can be found in [Bau93b]. The transformation technique itself will be described in a rather abstract way, and we will not supply a fully worked out algorithm here. Instead our description enables the design of several concrete algorithms, as long as they are within certain bounds (e.g. a *fairness* condition must be met).

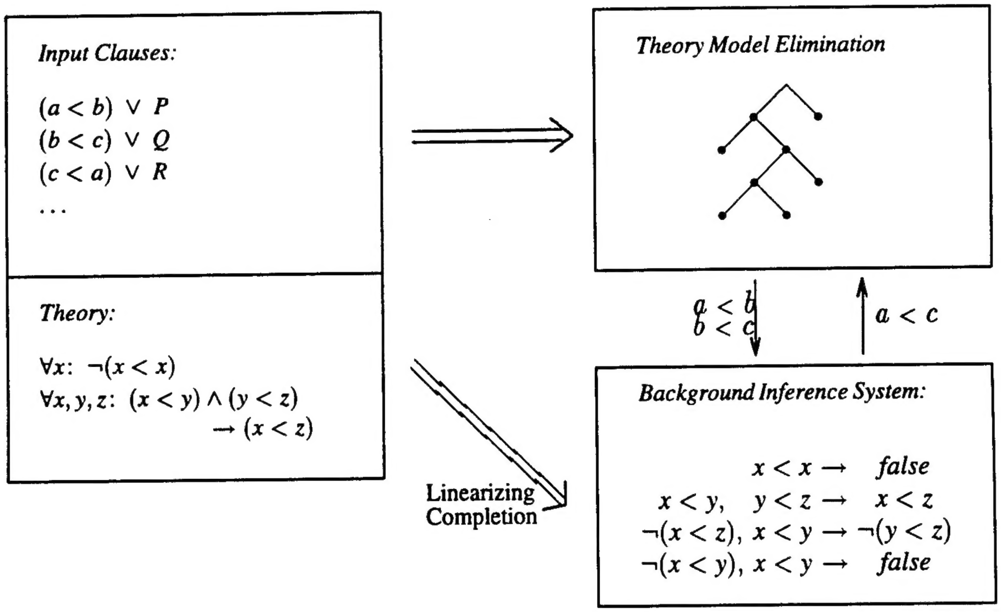

Input Clauses:
$(a < b) \lor P$
$(b < c) \lor Q$
$(c < a) \lor R$
...
Theory:
$\forall x\colon \neg(x < x)$
$\forall x, y, z\colon (x < y) \land (y < z) \rightarrow (x < z)$
Theory Model Elimination
$a < b$
$b < c$
$a < c$
Linearizing
Completion
Background Inference System:
$x < x \rightarrow false$
$x < y,\ y < z \rightarrow x < z$
$\neg(x < z),\ x < y \rightarrow \neg(y < z)$
$\neg(x < y),\ x < y \rightarrow false$

Figure 1: Application of *Linearizing Completion* within *partial theory model elimination.*

Our method works by saturating a Horn clause set $\mathcal{T}$ under several deduction operations until only redundant consequences can be added. The resulting (possibly infinite) system $\mathcal{I}_\infty(\mathcal{T})$ enjoys the following completeness property: for every (minimal) $\mathcal{T}$-unsatisfiable input literal set $\mathcal{L}$ and every literal $G \in \mathcal{L}$ there exists a *linear* resolution refutation of $\mathcal{I}_\infty(\mathcal{T}) \cup \mathcal{L}$ with goal literal G, i.e. G is processed stepwise until the empty clause is derived, *and*, all inferences are done in a *unit-resulting* way, i.e. in a n-literal parent clause at least $n - 1$ literals have to be simultaneously resolved against $n - 1$ complementary unit clauses in order to carry out an inference.

Thus our completion technique — called *linearizing completion* — is a device for combining the unit-resulting strategy of resolution [MOW76] with a linear strategy (à la Prolog, see [Llo84]) in a refutational complete way. This is not trivial, since although each of these strategies *alone* is complete for Horn theories, their naive combination is not. Furthermore we insist on completeness for *arbitrary* goal literal taken from the input set. All these properties are motivated by the intended application within linear theory reasoning calculi.

As an example consider the theory $\mathcal{T}$ of strict orderings which is axiomatized by the clauses

$$\mathcal{T} = \{\neg(x < x), \quad (x < y) \wedge (y < z) \rightarrow (x < z)\}$$

Our completion procedure produces the following finite set $\mathcal{I}_\infty(\mathcal{T})$ of clauses:

$$\mathcal{I}_\infty(\mathcal{T}): \boxed{\begin{array}{rcll} x < x & \rightarrow & \mathit{false} & \text{(Irref)} \\ x < y & \rightarrow & \neg(y < x) & \text{(Asym)} \\ x < y, y < z & \rightarrow & x < z & \text{(Trans-1)} \\ \neg(x < z), y < z & \rightarrow & \neg(x < y) & \text{(Trans-2)} \\ x < y, \neg(x < y) & \rightarrow & \mathit{false} & \text{(Syn)} \end{array}}$$

The associated *operational* meaning of, for instance, the clause (Trans-1) is "from literals $x < y$ and $y < z$ infer the literal $x < z$". Consequently, we call such clauses *inference rules*. Under this operational viewpoint it is clear that (Trans-1) and (Trans-2) are different, although they are logically equivalent. Now let $\mathcal{L}_1$ be a $\mathcal{T}$-unsatisfiable input literal set $\mathcal{L}_1 = \{\neg a < b, c < b, a < c\}$ In order to prove $\mathcal{L}_1$ as $\mathcal{T}$-unsatisfiable we can chain applications of the inference rules from $\mathcal{I}_\infty(\mathcal{T})$ to $\mathcal{L}_1$. If the goal literal $\neg a < b$ is choosen we find the following refutation R_1:

$$\neg a < b \xRightarrow{c<b}_{(\mathit{Trans}-2)} \neg a < c \xRightarrow{a<c}_{(\mathit{Syn})} \mathit{false}$$

Such a refutation consists of a sequence of inference steps which is terminated by *false*. Each inference step combines a so far derived literal (or, initially, the goal literal) with several given input literals according to the above mentioned operational meaning of inference rules.

Recall from above that we demand completeness for *arbitrary* goal literal. For instance, a refutation with goal literal $a < c \in \mathcal{L}_1$ should also exist. It is as follows (R_2):

$$a < c \xRightarrow{c<b}_{(\mathit{Trans}-1)} a < b \xRightarrow{\neg a<b}_{(\mathit{Syn})} \mathit{false}$$

The use of the (Trans-1) and of the (Trans-2) inference rules in the previous refutation should indicate that indeed both of them are needed.

Related Work. First we will mention some systems with *dedicated* theory reasoning components. A system for reasoning with (total) strict orderings was described in [Hin92]. In [BG93] it is demonstrated that the "chaining" inference rule of [Hin92] for transitive relations can be obtained by application of term-rewriting techniques within a more general resolution calculi. Another dedicated theory reasoning systems is *Z-Resolution* [Dix73], which builds in a theory consisting of two-literal clauses only. A more recent improvement was given in [Ohl90]. Finally, automatizing the theory of *equality* is a research topic of its own (e.g. "paramodulation" [RW69]).

In contrast to our approach, these inference systems are either tailored for one single theory, or are too restricted (the compilation of two-literal clauses only).

Other sources for related work are the completion techniques developed within the term-rewriting paradigm. In fact, our transformation technique was inspired by Knuth-Bendix completion [KB70] and its successors (e.g. [HR86, BDH86, Bac87]).

It is common to all these methods that they rely on *term-orderings* by which only certain — usually maximal — literals inside clauses may be selected for inferences. The refutations then are constructed from ordered inferences only. Here we see a major difference between these techniques and ours, as the *linearity* and the *unit-resulting* restrictions usually are not considered as a restriction for refutations from the completed theory. Consequently, the notion of a "goal" literal is also not a topic. An exception is [Ber90] which describes a procedure to complete towards a combination of term-ordering restrictions and the linear restriction. However, the unit-resulting restriction is not considered there.

2 Preliminaries.

Multisets are like sets, but allow for multiple occurrences of identical elements. We will use set-like notation with braces '$\{\!\!\{$' and '$\}\!\!\}$' for multisets. A *multiset with weight over a set X* is a tuple $\langle N, w\rangle$, also written as N_w (or $\{\!\!\{ \ldots \}\!\!\}_w$) where N is a multiset over X and $w \in \mathbf{N}$. Thus, a multiset with weight is obtained from an ordinary multiset by attaching some integer to. As a recursive generalization, define a *nested multiset with weight over a set X* as either an element from X or else as a multiset with weight over a nested multiset with weights. For instance, $\{\!\!\{a, \{\!\!\{b, c, \{\!\!\{d\}\!\!\}_3\}\!\!\}_4, e\}\!\!\}_2$ is such a set over $\{a, b, c, d, e\}$. They will be used as a complexity measure for proofs below. Using multisets as complexity measures is widely used (e.g. [Bac87]).

Furthermore we make heavily use of the data structure 'sequence'. If in the computations below a sequence appears where a multiset is required, the transformation from sequences to multisets is done in the obvious way.

Concerning substitutions we adopt the usual definitions (see e.g. [Llo84]). Substitutions are applied to multisets and sequences as expected. *Unification* is extended to multisets of literals as follows: A substitution σ is a unifier for N and M iff $N\sigma = M\sigma$. Multiset unification is of type "finitary". The elements of a complete set of unifiers are called MGUs.

We are mostly interested in Horn theories, which we formalize as follows: A *clause* is a multiset of literals, written as $L_1 \vee \cdots \vee L_k$. A *Horn clause* contains at most one positive literal. A *definite clause* contains exactly one positive literal. A *unit clause* contains exactly one literal. A *purely negative* clause contains only negative literals.

Concerning *(Herbrand) interpretation* we adopt the usual A *Horn theory* $\mathcal{T}$ is a satisfiable set of Horn clauses. A $\mathcal{T}$-interpretation is an interpretation which satisfies $\mathcal{T}$. Let M be a literal set or a clause set. Then M is called *$\mathcal{T}$-satisfiable* iff M is satisfied by some

$\mathcal{T}$-interpretation, otherwise M is called *$\mathcal{T}$-unsatisfiable.* It is easily verified that M is $\mathcal{T}$-unsatisfiable iff M is *false* in every $\mathcal{T}$-interpretation.

3 Inference Systems

Inference systems play the same role as *sets of rewrite rules* in Knuth-Bendix completion: they define inferences on the object-level. Inference systems are the objects of computation by *transformation systems* which are introduced in the next section.

This section introduces inference systems and basic properties such as "linearity" of proofs.

An *inference rule* (or *rule* for short) is a triple $P \rightarrow_w C$, where P is a multiset of literals, w is a non-negative integer and C is a literal. C may also be the "new" literal *false*, which is assumed to be distinct from all other literals. P is called the *premise*, w is called the *weight* and C is called the *conclusion* of the inference rule. In the sequel we will assume that w is always taken from some finite subset $W \subset \mathbf{N}$. In inference rules we will often write $L_1, \ldots, L_n \rightarrow_w C$ instead of $\{\!\{L_1, \ldots, L_n\}\!\} \rightarrow_w C$ and $P, Q \rightarrow_w C$ instead of $P \cup Q \rightarrow_w C$, etc. Also, the weight is often dropped if not relevant. The declarative reading of an inference rule is $L_1 \wedge \cdots \wedge L_n \rightarrow L_{n+1}$. The *weights* are motivated by the possibility for extended redundancy checks.

An *instance* of an inference rule is obtained by application of a substitution to both the premise and the conclusion. *Ground* inference rules do not contain variables. A *(ground) inference system* is a set of (ground) inference rules. If $\mathcal{I}$ is an inference system then $\mathcal{I}^g$ is defined as the inference system consisting of all ground instances of all rules from $\mathcal{I}$. We say that a literal *C' is inferred from a literal multiset P' by an inference rule $P \rightarrow_w C$ and substitution δ*, written as an *inference* $P' \Longrightarrow_{P \rightarrow_w C, \delta} C'$ iff $P' = P\delta$ and $C\delta = C'$. P' is also called the *premise*, C' is called the *conclusion* of the inference and $P \rightarrow_w C$ is called the *used* inference rule.

An *$\mathcal{I}$-derivation of L_n from input literals M with top literal L_1* and length $n \geq 0$ is a sequence

$$L_1 \overset{D_1}{\Longrightarrow}_{P_1 \rightarrow_{w_1} C_1, \delta_1} L_2 \overset{D_2}{\Longrightarrow}_{P_2 \rightarrow_{w_2} C_2, \delta_2} L_3 \cdots L_n \overset{D_n}{\Longrightarrow}_{P_n \rightarrow_{w_n} C_n, \delta_n} L_{n+1} \qquad (1)$$

such that for $i = 1, \ldots n$ the following holds: (1) D_i is a sequence $D_i^1 \cdots D_i^{m_i}$ of derivations of $L_i^1, \ldots, L_i^{m_i}$, respectively, from M with top literals from M, and (2) $P_i \rightarrow_{w_i} C_i \in \mathcal{I}$, and (3) there exists a substitution δ_i such that $L_i, L_i^1, \ldots, L_i^{m_i} \Longrightarrow_{P_i \rightarrow_{w_i} C_i, \delta_i} L_{i+1}$. Every sequence D_i is called *side derivation*, and its elements are written by juxtaposition; ϵ denotes the empty sequence. L_{n+1} is also called the *derived literal of D*.

Some more terminology is convenient: a derivation D is called *ground* iff the top literal and every of its inferences is ground; D is called *trivial* iff its length n is 1, i.e. iff no inferences are contained in D. D is called a *refutation* iff it is a derivation of *false*. An inference rule is said to be *used* in D iff (some instance of) it is used in some derivation step occurring in D. D is called *linear* iff every side derivation D_i is a – possibly empty

– sequence of trivial derivations, i.e. D_i is a sequence of literals. Otherwise it is called *non-linear.*

Note that a derivation does not instantiate the input literals. This is the same as in a "rewriting" proof in the term rewriting paradigm. Carrying on this analogy, a derivation relates to a first-order derivation much like "rewriting" relates to "narrowing".

As the first step of linearizing completion a given Horn theory $\mathcal{T}$ is re-written in a straightforward way as a set of inference rules. This transformation yields an *initial inference system,* $\mathcal{I}_0(\mathcal{T})$, and is defined as follows: (1) every positive unit clause A in $\mathcal{T}$ becomes an inference rule $\neg A \rightarrow \mathit{false}$ in $\mathcal{I}_0(\mathcal{T})$, (2) every definite clause $\neg A_1 \vee \cdots \vee \neg A_n \vee A_{n+1}$ becomes $A_1, \ldots, A_n \rightarrow A_{n+1}$, (3) every completely negative clause $\neg A_1 \vee \cdots \vee \neg A_n$ becomes $A_1, \ldots, A_n \rightarrow \mathit{false}$ and (4) for every n-ary predicate symbol p the rule $p(x_1, \ldots, x_n), \neg p(x_1, \ldots, x_n) \rightarrow \mathit{false}$ is added. Furthermore, some arbitrary chosen weight $w \in \mathbf{N}$ is attached to every rule in $\mathcal{I}_0(\mathcal{T})$

Example 3.1 Let $\mathcal{T} = \{\neg A \vee \neg B, \neg D \vee B, \neg C \vee A, B\}$ be a ground Horn theory. Then an initial inference $\mathcal{I}_0(\mathcal{T})$ system is

$$\boxed{\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl@{\qquad}rcl} A,B & \rightarrow_3 & \mathit{false} & D & \rightarrow_4 & B & C & \rightarrow_5 & A & \neg B & \rightarrow_6 & \mathit{false} \\ A,\neg A & \rightarrow_1 & \mathit{false} & B,\neg B & \rightarrow_1 & \mathit{false} & C,\neg C & \rightarrow_1 & \mathit{false} & D,\neg D & \rightarrow_1 & \mathit{false} \end{array}}$$

Next let $M_1 = \{D, C\}$. This is a derivation of *false* from M with top literal D (weights are omitted):

$$D_1 \quad = \quad D \overset{\varepsilon}{\Longrightarrow}_{D \rightarrow B} B \xrightarrow{C \overset{\varepsilon}{\Longrightarrow}_{C \rightarrow A} A}_{A,B \rightarrow \mathit{false}} \mathit{false}$$

D_1 is non-linear, since the seond inference step violates linearity. □

Initial inference systems constitute an "almost complete" calculus for the underlying theory. In order to explain this consider $M_2 = \{A\}$. Although M_2 is $\mathcal{T}$-unsatisfiable there is no $\mathcal{I}_0(\mathcal{T})$-refutation of M. In particular, the rule $A, B \rightarrow \mathit{false}$ cannot be applied since there is no input literal B. On the other hand a literal B is "hidden" in the theory and has been turned into a rule $\neg B \rightarrow \mathit{false}$. In order to obtain completeness it is necessary to access all positive unit literals, no matter whether they are contained in the input set or contained in the theory. Therefore we define for an inference system $\mathcal{I}$ the set $\mathit{punit}(\mathcal{I}) = \{A \mid \neg A \rightarrow \mathit{false} \in \mathcal{I}\}$ as the set of positive unit clauses from $\mathcal{I}$. Then completeness holds as follows:

Lemma 3.2 (Ground completeness of $\mathcal{I}_0(\mathcal{T})$) *Let $\mathcal{T}$ be a finite ground theory (i.e. a theory consisting of ground clauses only) and M be a finite set of ground literals. If M is $\mathcal{T}$-unsatisfiable then $L \overset{*}{\Longrightarrow}_{\mathcal{I}_0(\mathcal{T}), M \cup \mathit{punit}(\mathcal{I}_0(\mathcal{T}))}$ false for some $L \in M$.*

Proof. By definition of $\mathcal{T}$-unsatisfiability, M is $\mathcal{T}$-unsatisfiable iff $M \cup \mathcal{T}$ is unsatisfiable, where M is considered as a set of unit clauses. Since the unit-clauses of $\mathcal{T}$ can be used as input literals (via $\mathit{punit}(\mathcal{I}_0(\mathcal{T}))$) an existing Hyper-resolution refutation of $M \cup \mathcal{T}$ can be reflected in our framework. □

This completeness result is unsatisfactory, since (1) the literals $\mathit{punit}(\mathcal{I}_0(\mathcal{T}))$ are needed, and (2) a linear derivation may not always exist. For instance, there does not exists a linear refutation of M_1 in Example 3.1 with top literal D. We are going to tackle these problems next.

4 Orderings and Redundancy

The transformation systems defined below allow for the deletion of *redundant* inference rules. Redundancy in turn is based on orderings for derivations. Hence these notions have to be introduced before.

Let X be a set, well-founded wrt. the ordering $\succ$. As usual we define $s \prec t$ iff $t \succ s$, $s \preceq t$ iff $s \prec t$ or $s = t$ and $s \succeq t$ iff $t \preceq s$. Let $W \subseteq \mathbf{N}$ be a finite set of *weights.* A *multiset with weights over a set* X is a tuple $\langle N, w\rangle$, also written as N_w (or $\{\!\{ \dots \}\!\}_w$) where N is a multiset over X and $w \in W$. The ordering $\succ_{MW}$ on multisets with weights over X is defined as the lexicographical extension of $\succ$ and $>$, the natural ordering on integers. Thus we first compare the set-components; if these are equal then the weight gives the decision.

A *nested multiset with weights over* X is either an element from X or else is a multiset with weights over a nested multiset with weights. Orderings on (ordinary) multisets can be extended towards *nested multisets*[DM79]. For our purposes we have to go a little further and define the *nested multiset ordering with weights,* $\succ_{NMW}$, in the same way as the usual nested multiset ordering, except that multiset comparison is replaced by comparison of multisets with weights[1] Alternatively, $\succ_{NMW}$ can be defined as a recursive path ordering with status (see e.g. [Ste90]). For this, the multiset constructor "$\{\!\{ . \}\!\}$" is given a multiset status, and the tuple-constructor $\langle N, w\rangle$ (to attach weights to multisets) is given a left-right status.

Building on $\succ_{NMW}$ we will judge the "degree of linearity" of a derivation. For this let D be a derivation as exhibited in (1) above, and define the *complexity of D, compl*(D) as

$$compl(D) := \{\!\{0, \langle compl(D_1), w_i\rangle, \dots, \langle compl(D_n), w_n\rangle\}\!\}$$

where for a sequence $D^1 D^2 \cdots D^n$ of derivations we define

$$compl(D^1 D^2 \cdots D^n) := \bigcup_{i=1\dots n} compl(D^i) \ .$$

Thus, $compl(D)$ is a multiset whose elements are nested multisets with weights over the set $\{0\}$. $compl(D)$ contains structural information about the derivation: it expresses the shape (when read as a tree) of the derivation encoded as multisets, and occurrences of input literal are mapped to the dummy element 0 at the leafs. Furthermore, the weight of a used inference rules comes into the complexity measure as the weight of the multiset corresponding to the rule application. For instance, the derivation D_1 in example 3.1 has the complexity $\{\!\{0, \{\!\{\}\!\}_4, \{\!\{0, \{\!\{\}\!\}_5\}\!\}_3\}\!\}$.

In order to compare two derivations D_1 and D_2 we attach the artificial weight 0 to them and use the nested multiset ordering with weights. More formally we define now

$$D_1 \succ_{Lin} D_2 \text{ iff } \langle compl(D_1), 0\rangle \succ\!\!\succ_{NMW} \langle compl(D_2), 0\rangle$$

[1] We are aware that this definition is quite fuzzy; again, the details can be found in [Bau93a].

where the base set X is $\{0\}$ and the extended ordering is the empty ordering. Fortunately we have:

Proposition 4.1 *$\succ_{Lin}$ is a well-founded and monotonic ordering on derivations*[2].

Evidently, a derivation D is *linear* (cf. Section 3) iff its complexity $compl(D)$ is of the form $\{\!\{0, \{\!\{0, \ldots, 0\}\!\}_{w_1}, \ldots, \{\!\{0, \ldots, 0\}\!\}_{w_n}\}\!\}$. The ordering $\succ_{Lin}$ is defined in such a way that smaller derivations are "more linear".

The following definition of redundancy is defined slightly more general than needed:

Definition 4.2 **($\succ$-redundancy)** Let $\succ$ be a well-founded ordering on derivations (also called *derivation ordering* from now on), and let $\mathcal{I}$ be an inference system. An inference rule $P \rightarrow C$ is called *$\succ$-redundant in $\mathcal{I}$* ($\mathcal{I}$ need not necessarily contain $P \rightarrow C$) iff

1. $P \rightarrow C$ is not of the form $K \rightarrow$ *false* where K is a literal, and

2. for every inference system $\mathcal{J}$ with $\mathcal{I} \subseteq \mathcal{J}$ and every ground derivation
$$D = L_1 \Longrightarrow^{*}_{(\mathcal{J} \cup \{P \rightarrow C\})^g, M} L_n$$
which uses $P \rightarrow C$ there also exists a ground derivation $\mathcal{D}' \prec \mathcal{D}$ of the form
$$D' = L_1 \Longrightarrow^{*}_{(\mathcal{J} \setminus \{P \rightarrow C\})^g, M} L_n \quad .$$

□

$\succ$-redundancy means that any ground derivation in a possibly extended inference system $\mathcal{J}$ using the redundant inference rule can be replaced by a smaller derivation wrt. $\succ$ which uses at most the input literals as given, and this derivation does not use the redundant rule.

An inference rule of the form $K \rightarrow \mathit{false}$ is never redundant due to (1). The motivation for this comes from the use of such rules in the role of input literals in intermediate stages of completion (cf. also the proof of Lemma 3.2 above). It turns out that the deletion of a rule $K \rightarrow \mathit{false}$ in general does not provide a substitute for this purpose, even if the rule would be redundant according to condition 2.

In general, $\succ$-redundancy may be undecidable. However, $\succ$-redundancy plays a vital role in completion procedures. Thus at least a sufficient condition for $\succ$-redundancy should be given in a more constructive way. For linearizing completion we offer the following criterion:

Proposition 4.3 **(Sufficient $\succ_{Lin}$-redundancy criterion)** *Let $\mathcal{I}$ be an inference system and $P \rightarrow_w C$ be an inference rule. Suppose that for every $L \in P$ there exists a linear $\mathcal{I} \setminus \{P \rightarrow_w C\}$-derivation from P*
$$L \equiv L_1 \overset{D_1}{\Longrightarrow}_{P_1 \rightarrow_{w_1} C_1} L_2 \overset{D_2}{\Longrightarrow}_{P_1 \rightarrow_{w_2} C_1} L_3 \cdots L_{n-1} \overset{D_{n-1}}{\Longrightarrow}_{P_1 \rightarrow_{w_{n-1}} C_1} L_n \equiv C$$
with $n \geq 1$ and such that for $i = 1, \ldots, n-1$ it holds $\langle P \setminus \{\!\{L_1\}\!\}, w \rangle \succ\!\!\succ_{MW} \langle D_i, w_i \rangle$ In this comparison the sequence D_i of literals is to be read as a multiset. Then $P \rightarrow_w C$ is $\succ_{Lin}$-redundant in $\mathcal{I}$.

[2] An ordering is called *monotonic* iff replacement of a subderivation by a smaller one makes the whole derivation smaller.

Proposition 4.3 is valuable, since it gives us a more "local" criterion to detect redundancy than Definition 4.2. Informally, the condition $\langle P \setminus \{\!\{L_1\}\!\}, w\rangle \succ\!\!\succ_{MW} \langle D_i, w_i\rangle$ means that the i-the derivation step either uses strictly less side literals (the set D_i) than the side literals $P \setminus \{\!\{L\}\!\}$ of an inference step carried out with $P \rightarrow_w C$, or else, the side literals are the same, but an inference rule of smaller weight is used. This check has to be done for every $L \in P$ according to the potential applications of the inference rule in a derivation. We will not prove this proposition here, and note only that monotonicity of $\succ_{Lin}$ is crucial (cf. Proposition 4.1) for the proof.

Example 4.4 Consider the rule $R = x < x', x' < y', y' < z' \rightarrow x < z'$ which expresses a once unfolded transitivity rule $\mathit{Trans} = x < y, y < z \rightarrow x < z$. We claim that R is $\succ_{Lin}$-redundant in an inference system which contains *Trans*. Using proposition (4.3) this is checked as follows: for $x < x'$ as top literal consider the derivation

$$x < x' \xRightarrow{x'<y' \quad y'<z'}_R x < z'$$

whose side literals are $\{\!\{x' < y', y' < z'\}\!\}$. It can be replaced by the derivation

$$x < x' \xRightarrow{x'<y'}_{\mathit{Trans}} x < y' \xRightarrow{y'<z'}_{\mathit{Trans}} x < z'$$

Furthermore for the first derivation step it holds $\{\!\{x' < y'\}\!\} \subset \{\!\{x' < y', y' < z'\}\!\}$ and for the second step $\{\!\{y' < z'\}\!\} \subset \{\!\{x' < y', y' < z'\}\!\}$. Similarly there exist derivations with top literals $x' < y'$ and $y' < z'$. □

5 Transformation Systems

Transformation systems are the formal device for transformations of the *inference systems* of the previous section. A transformation systems performs a stepwise transformation of an initial inference system in a *fair* way by application of certain *transformation rules* into a *completed state*. Completed inference systems in turn are refutational complete wrt. the desired linearity and unit-resulting restrictions.

Transformation systems are rather general devices and allow for the construction of a wide range of restricted inference systems. In this paper we concentrate on the instance "linearizing completion".

Definition 5.1 (Transformation system) A *transformation rule* D with *premise* P and *conclusion* C is an expression of the form $\frac{P}{C}$ where P is a set of inference rules and C is an inference rule. A transformation rule can be labelled as *mandatory* or *optional* (cf. also Def. 6.1 below). A *transformation system* consists of a set of transformation rules and a well-founded ordering $\succ$ on derivations.

Let $\mathcal{D}$ be a transformation system. The relation $\mathcal{I} \vdash_{\mathcal{D}} \mathcal{I}'$ on inference systems means that $\mathcal{I}'$ is obtained from $\mathcal{I}$ by either (1) adding the conclusion of a transformation rule from $\mathcal{D}$ which can be applied to variable disjoint variants of rules in $\mathcal{I}$, or else (2) by deleting a $\succ$-redundant inference rule from $\mathcal{I}$. In case (1) the weight of C can be chosen arbitrary.

$$\textbf{Unit1:} \qquad \frac{L_1 \to C \qquad L_2 \to \mathit{false}}{\overline{C}\sigma \to \mathit{false}} \qquad \{\ \text{If } \overline{L_1}\sigma = L_2\sigma \text{ by MGU } \sigma$$

$$\textbf{Unit2:} \qquad \frac{L_1, P \to C \qquad L_2 \to \mathit{false}}{(P \to C)\sigma} \qquad \left\{ \begin{array}{lll} \text{If} & (1) & P \neq \emptyset, \text{ and} \\ & (2) & (\overline{L_1} = L_2)\sigma \text{ by MGU } \sigma \end{array} \right.$$

$$\textbf{Deduce:} \qquad \frac{P_1 \to C_1 \qquad L_2, P_2 \to C_2}{(P_1, P_2 \to C_2)\sigma} \qquad \left\{ \begin{array}{lll} \text{If} & (1) & P_2 \neq \emptyset, \text{ and} \\ & (2) & C_1\sigma = L_2\sigma \text{ by MGU } \sigma \end{array} \right.$$

$$\textbf{Contra:} \qquad \frac{L, P \to C}{\overline{C}, P \to \overline{L}} \qquad \{\ \text{If } C \neq \mathit{false}$$

Figure 2: The transformation rules of the transformation system *Lin*. Here, L and C are literals and P is a literal multiset.

A *$\mathcal{D}$-deduction from an inference system $\mathcal{I}_0$* is a sequence $\mathcal{I}_0 \vdash_{\mathcal{D}} \mathcal{I}_1 \vdash_{\mathcal{D}} \cdots \vdash_{\mathcal{D}} \mathcal{I}_n \vdash_{\mathcal{D}} \cdots$. Deductions may be of finite or infinite length.

The transformation system *Lin* consists of transformation rules given in Figure 2. The transformation rules **Unit1, Unit2** and **Deduce** are labelled as mandatory, and **Contra** is labelled as optional. As the derivation ordering we use $\succ_{\mathit{Lin}}$ as defined in section 4. □

Example 5.2 From the ground rules $A \to B$ and $\neg A \to \mathit{false}$ the rule $\neg B \to \mathit{false}$ can be obtained by **Unit1**[3]. Consider the inference system $\mathcal{I}_0(\mathcal{T})$ of example 3.1 again. From $A, B \to \mathit{false}$ and $\neg B \to \mathit{false}$ we can obtain $A \to \mathit{false}$ by **Unit2**. **Unit1** and **Unit2** share the same purpose: to eliminate in derivations applications of literals from *punit*($\mathcal{I}$) (these are initially needed, as stated in lemma 3.2). For instance, the set $M_2 = \{A\}$ from above now has a one-step refutation in $\mathcal{I}_0(\mathcal{T}) \cup \{A \to \mathit{false}\}$. From the rules $C \to A$ and $A, B \to \mathit{false}$ the rule $C, B \to_{10} \mathit{false}$ can be **Deduced**. Deduced rules are used to stepwisely turn a refutation into a "more linear" refutation. Again, using $\mathcal{I}_0(\mathcal{T}) \cup \{C, B \to_{10} \mathit{false}\}$ the refutation D_1 of $M_1 = \{D, C\}$ can be linearized with the new rule. We have:

$$D_1' \quad = \quad D \overset{\varepsilon}{\Longrightarrow}_{D \to B} B \overset{C}{\Longrightarrow}_{C,B \to \mathit{false}} \mathit{false}$$

The complexity of D_1' is $\{\!\!\{0, \{\!\!\{\}\!\!\}_4, \{\!\!\{0\}\!\!\}_{10}\}\!\!\}$. It can be verified that $D_1 \succ_{NMW} D_1'$. From the transitivity rule $x < y, y < z \to x < z$ and a copy $x' < y', y' < z' \to x' < z'$ by **Deduce** with $\sigma = \{y \leftarrow x', z \leftarrow z'\}$ one obtains $x < x', x' < y', y' < z' \to x < z'$. From $A, B \to C$ by **Contra** $\neg C, B \to \neg A$ can be obtained. □

This **Contra** transformation rule is an optional rule and thus is not labelled as *mandatory*. It is sometimes valuable in order to come to a finite system (the example at the end of section 5

[3] Weights are not necessary in this example in order to obtain the desired redundancy results.

makes use of the **Contra** rule). On the other side the **Contra** rule should be applied carefully since it increases the search space of the generated inference systems.

The process of applying the transformation rules of a transformation system may terminate or not. In order to treat both cases in a uniform way it is useful to define the *limit* inference system $\mathcal{I}_\infty$ which is finite if the transformation system eventually does not produce new inference rules any more and infinite otherwise.

Definition 5.3 (Limit, [Bac87]) The *limit of a deduction* $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots \vdash \mathcal{I}_n \cdots$ is defined as $\mathcal{I}_\infty := \bigcup_i \bigcap_{j \geq i} \mathcal{I}_j$. The elements of $\mathcal{I}_\infty$ are also called the *persisting* inference rules. $\mathcal{I}_\infty$ is the set of inference rules generated eventually and never deleted afterwards. □

We find that the following central property holds:

Lemma 5.4 *Let $\mathcal{D}$ be a transformation system with derivation ordering $\succ$. Let $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ be a $\mathcal{D}$-deduction. If there exists a derivation $D = L \overset{*}{\Longrightarrow}_{\mathcal{I}^g_k, M \cup punit(\mathcal{I}^g_k)} L'$ then there also exists a derivation $D' = L \overset{*}{\Longrightarrow}_{\mathcal{I}^g_\infty, M \cup punit(\mathcal{I}^g_\infty)} L'$ with $D' \preceq D$.*

6 Fairness and Saturation

Deductions must be *fair*, which roughly means that no application of a mandatory transformation rule is deferred infinitely long. Fairness is important since it entails that "enough" inference rules to obtain normal derivations are generated. Our definition of fairness is an adaption of standard definitions in the term-rewriting literature (see e.g. [Bac87]).

Definition 6.1 (Fairness) Let $\mathcal{D}$ be a transformation system with derivation ordering $\succ$. A $\mathcal{D}$-deduction $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots \vdash \mathcal{I}_n \cdots$ is called *fair* iff whenever $\mathcal{I}_\infty \vdash \mathcal{I}_\infty \cup \{P \to C\}$ for some application of a mandatory transformation rule from $\mathcal{D}$, then for some k, $P \to C \in \mathcal{I}_k$ up to renaming, or $P \to C$ is $\succ$-redundant in $\mathcal{I}_k$. □

Fairness states that it is sufficient either to generate an inference rule or to prove it redundant from persisting inference rules only. This notion of fairness enables the use of a "delete as many inference rules as possible" strategy in implementations, since a rule once shown to be redundant is redundant in all subsequent stages and thus need not persist.

The next central concept is *saturation* which means that only trivial new inference rules can be generated from an inference system. Saturation is a useful concept since it allows to characterize refutational complete inference systems, which is a semantical concept, in a more syntactical way.

Definition 6.2 (Saturation) Let $\mathcal{D}$ be a transformation system with derivation ordering $\succ$. An inference system $\mathcal{I}$ is *completed (wrt. $\mathcal{D}$)* iff whenever $\mathcal{I} \vdash_{\mathcal{D}} \mathcal{I} \cup \{P \to C\}$ by application of a mandatory transformation rule from $\mathcal{D}$ then $P \to C \in \mathcal{I}$ up to renaming or $P \to C$ is $\succ$-redundant in $\mathcal{I}$. □

Fairness, deductions and saturation relate as follows:

Theorem 6.3 *Let $\mathcal{D}$ be a transformation system and $\mathcal{I}_0$ be an inference system. The limit $\mathcal{I}_\infty$ of a fair $\mathcal{D}$-deduction $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ is completed wrt. $\mathcal{D}$.*

Up to now we have seen that the inference systems generated along a deduction never increase the complexity of a once obtained derivation. However, in order to obtain completeness wrt. normal-form derivations (linear derivations) more is required: the transformation system has to eventually generate inference rules that will strictly *de*crease the complexity of a non-normal derivation. Since the **deduce** transformation rule is labelled as mandatory, and furthermore works towards strictly decreasing derivations wrt. the well-founded ordering $\succ_{Lin}$, we can linearize every refutation in a finite way. However, such linear a refutation might still use unit inference rules being turned via $punit(\mathcal{I})$ into input literals. However, every use of a literal from $punit(\mathcal{I})$ represents a computation among the inference rules and should be avoided. In order to admit refutations with input literals M alone, we will compile the literals $punit(\mathcal{I})$ into the inference system. This is realized by means of the **Unit1** and **Unit2** transformation rules. Since these transformation rules are labelled as mandatory, and furthermore work towards strictly decreasing derivations wrt. the well-founded ordering $\succ_{Lin}$, we can in every refutation eliminate the usages of elements from $punit(\mathcal{I})$ in a finite way.

In order to formulate completeness we need one more notion:

Definition 6.4 (Relative Completeness) An inference system $\mathcal{I}$ is called *relative complete wrt. an inference system $\mathcal{J}$* iff whenever $L \Longrightarrow^*_{\mathcal{J}^g, M \cup punit(\mathcal{J}^g)} L'$ then also $L \Longrightarrow^*_{\mathcal{I}^g, M \cup punit(\mathcal{I}^g)} L'$
□

Completeness reads in its most general form as follows:

Theorem 6.5 (General Ground Completeness Theorem) *Let $\mathcal{T}$ be a theory. Suppose an inference system $\mathcal{I}$ completed wrt. Lin is relative complete wrt. $\mathcal{I}_0(\mathcal{T})$. Then for every $\mathcal{T}$-unsatisfiable ground literal set M there exists a linear $\mathcal{I}^g$-refutation with some top literal from M.*

This theorem requires the existence of a completed and relative complete inference system wrt. $\mathcal{I}_0(\mathcal{T})$. Such a system can be obtained in a constructive way as follows:

Proposition 6.6 *The limit $\mathcal{I}_\infty$ of a $\mathcal{D}$-deduction $\mathcal{I}_0(T) \vdash_{\mathcal{D}} \mathcal{I}_1 \vdash_{\mathcal{D}} \mathcal{I}_2 \cdots$ is relative complete wrt. $\mathcal{I}_0(\mathcal{T})$, where $\mathcal{T}$ is a theory.*

Proof. Use lemma 5.4, setting there $\mathcal{I}_0 = \mathcal{I}_0(\mathcal{T})$ and $k = 0$. □
By this proposition and by Theorem 6.3 we can instantiate Theorem 6.5:

Corollary 6.7 *Let $\mathcal{I}_\infty$ be the limit of a fair Lin-deduction starting with $\mathcal{I}_0(\mathcal{T})$. Then for every $\mathcal{T}$-unsatisfiable ground literal set M there exists a linear $\mathcal{I}^g_\infty$-refutation of M with some top literal from M.*

By the last theorem, proposition and the corollary we have established the main results of our saturation technique. It turns out that Theorem 6.5 and Corollary 6.7 can be even strengthened towards *arbitrary* top literals $K \in M$, provided that K is contained in some minimal $\mathcal{T}$-unsatisfiable subset of M. This property is important when a completed inference system is to be used as a background reasoner for theory reasoning.

Equivalence:

$$\begin{array}{rll} x = y, \neg x = y \rightarrow & \mathit{false} & \text{(Syn=)} \\ \neg x = x \rightarrow & \mathit{false} & \text{(Ref)} \\ x = y \rightarrow & y = x & \text{(Sym=-1)} \\ \neg x = y \rightarrow & \neg y = x & \text{(Sym=-2)} \\ x \doteq x', \; x = y \rightarrow & x' = y & \text{(Trans=-1-1)} \\ y \doteq y', \; x = y \rightarrow & x = y' & \text{(Trans=-1-2)} \\ x \doteq x', \neg x = y \rightarrow & \neg x' = y & \text{(Trans=-2)} \end{array}$$

Strict order:

$$\begin{array}{rll} x < y, \neg x < y \rightarrow & \mathit{false} & \text{(Syn<)} \\ x < x \rightarrow & \mathit{false} & \text{(IRef)} \\ x < y \rightarrow & \neg y < x & \text{(ASym<-1)} \\ x = y \rightarrow & \neg x < y & \text{(IRef-1)} \\ x' < x, \; x < y \rightarrow & x' < y & \text{(Trans<-1)} \\ x < x', \neg x < y \rightarrow & \neg x' < y & \text{(Trans<-2)} \end{array}$$

<-Substitution:

$$\begin{array}{rl} x \doteq x', \quad x < y \rightarrow & x' < y \\ x \doteq x', f^i(x) < y \rightarrow & f^i(x') < y \\ y \doteq y', \quad x < y \rightarrow & x < y' \\ y \doteq y', x < f^i(y) \rightarrow & x < f^i(y') \\ x \doteq x', \quad \neg x < y \rightarrow & \neg x' < y \\ x \doteq x', \neg f^i(x) < y \rightarrow & \neg f^i(x') < y \\ y \doteq y', \quad \neg x < y \rightarrow & \neg x < y' \\ y \doteq y', \neg x < f^i(y) \rightarrow & \neg x < f^i(y') \end{array}$$

f-Substitution:

$$\begin{array}{rl} x \doteq x', f^i(x) = y \rightarrow & f^i(x') = y \\ y \doteq y', x = f^i(y) \rightarrow & x = f^i(y') \\ x \doteq x', \neg f^i(x) = y \rightarrow & \neg f^i(x') = y \end{array}$$

Figure 3: A linear complete inference system for the theory $\mathcal{ES}$. Inference rules containing $f^i(x)$ have to be replaced by all i-fold instances $f(f(\cdots f(x)))$, $i > 0$; a dotted equation $x \doteq y$ is a nondeterministical notation for $x = y$ or $y = x$. Rules containing such equations have to be expanded for both versions.

7 Example

As a non-trivial example consider the joint theory $\mathcal{ES}$ of equality and strict orderings. The predicate symbol $=$ is interpreted as an equivalence relation and the predicate symbol $<$ is interpreted as a strict ordering (i.e. as a transitive and irreflexive relation). Furthermore we have include a single function symbol f of arity 1, which gives rise for the substitution axiom $\forall x : x = y \rightarrow f(x) = f(y)$. The extension to a richer signature is straightforward. Applying a fair Lin-deduction to the initial system $\mathcal{I}_0(\mathcal{ES})$ results in the (infinite) saturated inference system depicted in Figure 3. Note that not all contrapositives of the (Trans<) axiom have to be generated. We think that this inference system generalizes the standard inference rules of the linear paramodulation calculus (see e.g. [FHS89]) in a nice and useful way. Finite approximations of this system can be obtained in a fully automatical way by our implementation. By considering only finite subsets of the infinite system, we have a means for bounding the resources allocated for a proof attempt. For instance, we can control the maximal depth of a term which shall be considered for a paramodulation step.

8 Conclusions

In this paper we have developed a new completion technique for Horn theories that allows for the combination of the linear and unit-resulting restrictions. The central operation is to add new inference rules that detour violations of the linearity restrictions in unit-resulting refutations. A fairness condition guarantess completeness. The whole method was described in a rather abstract and non-deterministical way. Due to this abstract description a whole class of implementations can be described at a time. Indeed we have implemented the method and succesfully combined with a model elimination prover (this is not demonstrated in this paper).

The method is general enough to be applicable to other ordering criterion; for example one might think of ordering restrictions as applied in the term rewriting paradigm, or the combination of ordering restrictions and linearity restrictions. Of course, different restrictions require different transformation systems, but many of the concepts and claims not related to a specific restriction can be kept.

Acknowledgements: I thank J. Dix, U. Furbach, O. Menkens and F. Stolzenburg for reading earlier drafts of this paper.

References

[AS92] Owen L. Astrachan and Mark E. Stickel. Caching and Lemmaizing in Model Elimination Theorem Provers. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction (CADE-11)*, pages 224–238. Springer-Verlag, June 1992. LNAI 607.

[Bac87] Leo Bachmair. *Proof Methods for Equational Theories*. PhD thesis, University of Illinois at Urbana, U.S.A., 1987.

[Bau92] P. Baumgartner. A Model Elimination Calculus with Built-in Theories. In H.-J. Ohlbach, editor, *Proceedings of the 16-th German AI-Conference (GWAI-92)*, pages 30–42. Springer, 1992. LNAI 671.

[Bau93a] P. Baumgartner. Linear Completion: Combining the Linear and the Unit-Resulting Restrictions. Research Report 9/93, University of Koblenz, 1993.

[Bau93b] P. Baumgartner. Refinements of Theory Model Elimination and a Variant without Contrapositives. Research Report 8/93, University of Koblenz, 1993. (to appear in Proc. ECAI 94).

[BDH86] Leo Bachmair, Nachum Dershowitz, and Jieh Hsiang. Orderings for equational proofs. *IEEE*, pages 346–357, 1986.

[Ber90] Hubert Bertling. Knuth-Bendix Completion of Horn Clause Programs for Restricted Linear Resolution and Paramodulation. In S. Kaplan and M. Okada, editors, *Proceedings of the 2nd International Workshop on Conditional and Typed Rewriting Systems*, pages 181–193. Springer-Verlag, June 1990. LNCS 516.

[BF93a] P. Baumgartner and U. Furbach. Model Elimination without Contrapositives and its Application to PTTP. Fachbericht Informatik 12/93, Universität Koblenz, 1993. (short version in Proc. CADE-12).

[BF93b] P. Baumgartner and U. Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *I*nterface. (to appear in Proc. CADE-12), 1993.

[BG93] L. Bachmair and H. Ganzinger. Rewrite Techniques for Transitive Relations. Research Report MPI-I-93-249, Max-Planck-Institut für Informatik, 1993.

[Dix73] J. Dixon. Z-Resolution: Theorem-Proving with Compiled Axioms. *Journal of the ACM*, 20(1):127–147, 1973.

[DM79] N. Dershowitz and Z. Manna. Proving Termination with multiset orderings. *Comm. ACM*, 22:465–476, 1979.

[FHS89] Ulrich Furbach, Steffen Hölldobler, and Joachim Schreiber. Horn equational theories and paramodulation. *Journal of Automated Reasoning*, 3:309–337, 1989.

[Hin92] L. Hines. The Central Variable Strategy of Str±ve. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction (CADE-11)*, pages 35–49. Springer-Verlag, June 1992. LNAI 607.

[HR86] J. Hsiang and M. Rusinowitch. A New Method for Establishing Refutational Completeness in Theorem Proving. In *Proc. 8th CADE*, pages 141–152. Springer, 1986.

[KB70] Donald E. Knuth and B. Bendix, Peter. Simple world problems in universal algebras, 1970.

[Llo84] J. Lloyd. *Foundations of Logic Programming*. Springer, 1984.

[Lov68] D. Loveland. Mechanical Theorem Proving by Model Elimination. *JACM*, 15(2), 1968.

[Lov78] D. Loveland. *Automated Theorem Proving - A Logical Basis*. North Holland, 1978.

[LSBB92] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performace Theorem Prover. *Journal of Automated Reasoning*, 8(2), 1992.

[MOW76] J. McCharen, R. Overbeek, and L. Wos. Complexity and related enhancements for automated theorem-proving programs. *Computers and Mathematics with Applications*, 2:1–16, 1976.

[Ohl90] H.-J. Ohlbach. Compilation of Recursive Two-Literal Clauses into Unification Algorithms. In V. Sgurev Ph. Jorrand, editor, *Artificial Intelligence IV – Methodology, Systems, Applications*. Norh Holland, 1990.

[RW69] G. A. Robinson and L. Wos. Paramodulation and Theorem Proving in First Order Theories with Equality. In Meltzer and Mitchie, editors, *Machine Intelligence 4*. Edinburg University Press, 1969.

[Ste90] J. Steinebach. Improving Associative Path Orderings. In M.E. Stickel, editor, *Proc CADE 10, LNCS 449*, pages 411–425. Springer, 1990.

[Sti85] M.E. Stickel. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

Knowledge Modeling for Evolutionary Program Synthesis

Jutta Eusterbrock
Institute for Tele - Cooperation Technology
Rheinstr. 75, 64295 Darmstadt, Germany
eusterbr@darmstadt.gmd.de

Abstract

This paper proposes a multi-layer methodology - called SEAMLESS - for the design of knowledge-based program synthesis systems within the logic programming paradigm. The intent is to provide an integrated logical framework for modeling the different kinds of knowledge involved during program synthesis processes and a workbench of inference-based generic tools for the constructive solution of knowledge acquisition and program synthesis tasks. As major requirements, evolutionary synthesis processes, the re-usability of knowledge chunks, and multiple views should be facilitated. The approach depends upon the definition of hierarchically organized abstraction layers. It is a combination of abstract data types and typed predicate logic used for domain modeling. Metatheories allow for the specification of generic problem solving knowledge and generic tactics in a domain independent way. The interaction between theories and abstract representations is realized by interface definitions. Our approach borrows concepts from object-oriented programming to structure the comprehensive theory by frame-like types and to support inheritance. Meta-programming techniques provide language facilities to describe the dynamic change of theories and allow reasoning about theories. Based on this theoretical foundation, we implemented a knowledge based program synthesis system and applied it to the discovery of a non-trivial mathematical algorithm.

1 MOTIVATION

"A programming environment suitable for prototyping would support a design methodology that emphasizes creativity, experimentation, learning and evolution." ([8]), S. 13)

"Successful discovery requires the representation and manipulation of quantities of information, the efficient storage and retrieval of that information, and a processing strategy that focuses upon relevant segments of that information and pursues it with resources appropriate to the information's significance." ([5], S. 7)

The "proofs-as-programs" paradigm, ie. assuming declarative, first order domain specifications, proving goal specifications in a constructive manner by (inductive) proofs and elaborating algorithms and programs from proofs, has been proposed in the deductive community (cf. [24]) as a theoretical basis for program synthesis. The deductive approach covers only parts of real software development processes (cf. [16]). In human problem solving, the ability to make an appropriate abstraction is an important means for managing the complexity of problem-solving. To control program synthesis processes, formal specifications of problem solving principles, eg. "Global-Search" [20], are implemented. Experiences gathered so far from automatically solved complex algorithmic problems[1] suggest that computer-assisted solving of real problems involves - besides the guidance by abstract principles - comprehensive domain specific and strategic knowledge. Due to changing environments or missing information, requirement specifications are seldom precise and complete. Thus, specifications and implementations are subject to continual changes. As one of the few approaches that take this aspect in account, Shapiro's debugging method for Prolog programs [19] provides a means for the incremental, non-monotonic acquisition of requirement specifications.

Most of the devised techniques, which may faciliate specification and program construction processes, are analyzed in isolated contexts and depend on different formal languages. Generally, the deductive approach based on first order logic seems to be inadequate for organizing large knowledge bases and to describe the dynamic change of knowledge entities. To deal with these problems, numerous extensions of first order logic as specification languages have emerged. FRORL [22] is a object-oriented specification language, based on a non-monotonic variant of horn clause logic. FOOPS [9] unifies object-oriented and functional programming and provides generic modules. Gödel [10] is a typed logic programming language, which provides metaprogramming facilities. Programs are understood as theories which can be manipulated. While these formal languages are intended to be generally applicable, the domain modeling and the implementation of methods remain ad-hoc processes, which have to be done manually from scratch in each case.

Consequently, a methodology is wanted, which allows to apply a single notation and a closed concept for domain modeling as well as for knowledge acquisition and system construction. In the subsequent sections - rather than to propose a new formal language - we will show that various useful forms of knowledge, eg. domain data, strategic knowledge, and reasoning, eg. theory formation, for program synthesis tasks can be modeled and implemented within a coherent multi-layer logic-based framework.

The approach bridges the gaps between totally domain driven problem solving, abstract methods devised to support program synthesis processes and general theorem proving methods, and is semantically well understood. It provides the formal basis for the evolutionary development and implementation of a program synthesis environment, which integrates a workbench of tools and supports the program synthesis methodology SEAMLESS (**S**pecification, **E**xperimentation, **A**bstraction,

[1] Eg. the non-existence of a projective plane of order 10 [14]; the computer-assisted proof of a optimal depth lower bound for nine-input sorting networks [18].

Modification, Logical Validation, Extraction for System Synthesis). We implemented a prototype system, which presently supports a significant subset of the envisaged system.

The paper is organized as follows. In the following section, a brief outline of the knowledge representation principles is given. In the subsequent sections, formal concepts for modeling and reasoning about domain data and theories, domain abstractions and generic methods, and, dynamic changes and structured sets of theories are presented. Finally, this approach is summarized and compared with some other recent proposals for formalizing problem solving knowledge.

2 A FRAMEWORK FOR KNOWLEDGE BASED PROGRAM SYNTHESIS

A multiple meta-level architecture and a uniform knowledge representation language seem to be indispensable (cf. [13, 17, 23]) for integrating different knowledge sources and reasoning procedures. The SEAMLESS framework to model program synthesis knowledge is based upon the definition of three hierarchically organized abstraction layers. The different layers refer to the description of

- domain knowledge;
- generic domain abstractions;
- use and manipulation of knowledge sources.

Layers are abstractions from lower layers and may be regarded as separated knowledge bases associated with distinguished specification languages and interpreters for expressions. The communication among layers is realized by *interfaces.* Two views are possible: a layer may be seen as meta-layer, controlling the behaviour of the lower (object) layers, as well as an object layer which is available to manipulation by higher layers and provides services to them.

As is well known in logic programming, data types are useful for a lot of reasons. Type specifications can be replaced easily by more efficient ones, are easy extensible and can be used in more than one context. It is possible to encode the structure of the domain or additional knowledge directly into the terms, which results in efficient proof procedures. Some of the non-logical constructs in declarative programming can be avoided by employing data types. Type checking procedures may ensure that the formulas in the program are correct with respect to the type declarations. Hence, each layer is sub-divided. Data Type definitions are separated from the logic component and handled in a distinguished data type library. Our principle data types on the different abstraction layers are based on terms, formulas and - as a major extension to related approaches in knowledge based reasoning - theories.

For knowledge representation we employ a Gödel (cf. [10])-like typed logic language, suitably adapted to the specific representation tasks, extended by metalogic features and assumed to admit efficient and terminating proof procedures. Typed logic programs provide a means to link object and metalevel. They consist of data type definitions and typed first order theories. A declarative initial semantics is assumed. Tasks to be solved are specified as goals, ie. theorems to be proven. Solutions are generated by computation, ie. constructive theorem proving.

Further, we propose to separate knowledge-based program synthesis systems into hierarchically organized knowledge sources, called *modules*. Each modul is treated as an object, which is identified by a logical name and contains a chunk of knowledge. According to the various functionalities of knowlege, there are distinct kinds of modules, eg. *data types* or *interfaces.* A module is declared by its functionality, its name and the relations to associated modules.

3 DOMAIN KNOWLEDGE

The following section illustrates the application of typed logic programs at the domain knowledge level.

3.1 ABSTRACT DATA TYPES

Domain data is given by (real-world) objects, relations among objects, properties of objects and operations defined for the domain.

Domain Data is specified by means of a *First Order Data Type*, which consists of a *Signature*, which is composed of the set `sorts` of sorts together with a subsumption relation $\sqsubset$, the function domain which is disjunctly divided in constants, the constructor symbols and defined operation symbols, the variables, and the function declarations $\mathtt{f} : \mathtt{s}_1 \times \ldots \times \mathtt{s}_r \rightarrow \mathtt{s}$, $\mathtt{s}_1, \ldots, \mathtt{s}_r, \mathtt{s} \in \mathtt{sorts}$; *Constraints* which are first-order formulas and describe structural dependencies between objects; *Conditional Equations* which express behaviour of objects and functions.

We assume a declarative initial algebra semantics. Sorts are interpreted as subsets - called types - of the initial term algebra. The equations define an equivalence relation on the set of terms which is associated with a specific sort. To obtain efficient implementations of operations and proof procedures, canonical term representations for objects are assumed. Thus, functions may be implemented by rewriting terms.

Basic sorts are `atom`, `nat` or `bool`. Complex sorts are defined from simpler ones by applying *sort constructors* and *disjunction* ($\sqcup$) using the $\doteq$ operator. Commonly used inductive constructors are `list` (eg. `nat_list` $\doteq$ `list[nat]`), `sequence`, and the binary operator .. which constructs a term, given its root and a list of subterms. Functions may be defined in a generic way for values of distinct sorts and may cause similar effects. Reserved identifiers are: `destruct` to access individual components of terms; `replace` to replace distinguished subterms; `pretty-print` to visualise the structure of bulky terms; `display` to pars the internal representation of objects and visualise them in a user-understandable form.

3.2 DOMAIN THEORIES

The domain layer contains declarative knowledge about the application domain.

A *Domain Theory* is specified by a typed logic program. A *typed logic program* consists of first order data types; a signature and axioms, which are extended clauses. Let `A` denote an atom, `B` be a multiset of atoms, `C` be a multiset consisting of atoms and expressions of the form `f(X)` = `Y`, where `f` denotes an operator symbol. An *extended clause* is of the form `A` ⇐ `B` or `A` ⇐ `C`. A *goal* (or *query*) is of the form ⇐ `B` or ⇐ `C`. A *proof-tree* of a goal is a tree, whose leaves are axioms, whose nodes are proofs of deduced subgoals, and whose root is an instance of *Goal.*

A declarative, model theoretic semantics is assumed, eg. a domain theory is a correct specification, if the domain model is isomorphic to the intended model.

Example: The domain theory below defines topological sorting problems (cf. [12]), taking for granted that a data type `poset` is implemented.

```
Domain Declaration                 TopSort
Imports                            Poset

Sorts
poset, domain, bool

Predicates
topsort :                          poset × poset → bool
new_rel :                          poset × (domain × domain) → bool

Axioms
topsort(P,P)                   ⇐   totally_ordered(P).
topsort(P,S)                   ⇐   new_rel(P,[X,Y])  ∧  reduct(P,[X,Y]) = PNew
                                   ∧  topsort(PNew,S).
new_rel([Dom,Set_Rel],[X,Y])   ⇐   X ∈ Dom  ∧  Y ∈ Dom
                                   ∧  (X < Y) ∉ Set_Rel  ∧  (X > Y) ∉ Set_Rel.
reduct(P,[X,Y]) = PNew         ⇐   X < Y  ∧  PNew = P ∪ (X < Y).
reduct(P,[X,Y]) = PNew         ⇐   X > Y  ∧  PNew = P ∪ (X > Y).
```

4 GENERIC DOMAIN ABSTRACTIONS

The representation of domain knowledge and goals in a standard (typed) first-order language and realizing proofs by resolution is unsatisfactory. Most often the specifications are too hard to be proven automatically, although there seems to be a natural way of goal-oriented reasoning. For instance, a statement is to be proven by transforming it to an equivalent statement and applying an already proven theorem. Generic domain theories are a means to represent strategic knowledge in a domain independent abstract form. Generic methods offer the possibility of combining deductive proof procedures with specialized problem dependent solution procedures.

4.1 GENERIC DOMAIN THEORIES

A major characteristic of our design model is the distinction between *domain theories* and a *generic domain theories* - common descriptions of collections of *domain theories.* A generic domain theory is a second order theory where first-order formulas are subject to manipulation and reasoning by well-defined procedures. As before, we separate data types from the logic component.

A *generic domain type* is defined by a signature and axioms. Sorts are identified with a subset of the class of first order formulas and terms. New sorts are defined from already defined ones by schematic expressions that describe the specific form of the expressions belonging to the subtype, employing the constructors $\forall \mathtt{U}, \exists \mathtt{U}$, where U is an arbitrary variable and $\Rightarrow$. Specification formulas and proof terms are introduced by the expressions $\mathtt{spec} \doteq \forall \mathtt{U}[\exists \mathtt{Z} : \mathtt{prop}_{\mathtt{in}}(\mathtt{U}) \Rightarrow \mathtt{prop}_{\mathtt{out}}(\mathtt{U}, \mathtt{Z})]$ and $\mathtt{proof} \doteq [] \sqcup ..(\mathtt{spec}, \mathtt{list}[\mathtt{spec}])$. As in the case of first-order datatypes, *destructors* access the components of complex datatypes by names; eg.
destruct(pre_arg, Spec, Arg) accesses the argument in the precondition of Spec and formulas may be rewritten by the application of *replacement* functions.

Generic Domain Theories describe the syntactical structure and semantical properties of a class of domain theories in an uniform way. A *Generic Domain Theory* is defined by a typed higher order logic, which consists of generic data types; a signature and axioms. Predicate symbols may be declared as *parametric.* Its intended semantics is stated by soundness axioms.

As a concrete example for a generic theory, *Decomposition Theories* are introduced. Decomposition Theories capture the essence of the class of domain theories, which are amenable to be analyzed by the well known "Divide-and-conquer"-Paradigm.

Domain Type	**DecompositionTheory**
Imports	**Formula**
Sorts	
term,spec,proof	
ParametricMetapredicates	
provable :	**spec → bool**
<spec:	**spec × spec → bool**
minimal_spec :	**spec × proof → bool**
split_spec :	**spec × term → bool**
decompose_spec :	**spec × term × spec × spec → bool**
know :	**spec × proof → bool**
behav_equiv :	**spec × spec → bool**

The intended semantics of the parametric metapredicates will be given informally.

provable(Spec) is intended to be true, iff Spec is derivable.

$<_{\mathrm{spec}}$ is intended to be a well-founded ordering on specifications and minimal_spec verifies the minimality condition.

decompose_spec(Spec, X, Spec′, Spec″) splits Spec into Spec′ and Spec″, applying a function X, which is computed calling split_spec(Spec, X). Spec is derivable iff Spec′ and Spec″ are derivable.

know(Spec, Proof) means that Spec is provable and Proof denotes its proof tree.

behav_equiv(Spec, SpecEq) is intended to be true, iff SpecEq is a specification, constructable from Spec with the property provable(SpecEq) iff provable(Spec).

4.2 CONSTRUCTIVE GENERIC METHODS

The program synthesis task we consider can be formulated as follows: Given a declarative domain theory, prove the (non)-existence of a deterministic logic program which proves all instances of a set of goals, which could be derived from the domain theory and whose worst-case depth is bounded by a given number. We propose to design constructive generic methods.

A Deductive, Constructive Generic Method is a typed higher order definition of a metapredicate **method** : **spec** × **proof** × ... → **bool** that declaratively represents the derivability of specifications with type **spec** for all admissible interpretations of a stated generic domain theory and supplements a constructive solution of type **proof**. Further requirements as depth bounds may be specified by supplementary arguments of **method**.

The figure below gives an example of a generic method for solving the above stated program synthesis task, extending the *Divide-and-Conquer* paradigm by allowing strategic knowledge to be employed for the control of proof processes.

```
Generic Method                       Knowledge Based Divide-and-Conquer
Imports                              Interface(-)

Sorts
spec, proof, nat

Metapredicates
div_and_conq                         spec × proof × nat → bool
compose_proof                        proof × proof × nat → proof

Axioms
div_and_con(Spec,Proof,Depth)    ⇐   know(Spec,Proof,DepthK)  ∧  DepthK ≤ Depth.
div_and_con(Spec,Proof,Depth)    ⇐   behav_equiv(Spec,SpecBeh,ProofB)  ∧  <spec (Spec,SpecBeh)
                                     ∧ know(SpecKnow,ProofK,DepthK)  ∧  DepthK ≤ Depth
                                     ∧ compose_proof(ProofB,ProofK,Proof).
div_and_con(Spec,[],Depth)       ⇐   minimal_spec(Spec)  ∧  Depth ≥ 0.
div_and_con(Spec,Proof,Depth)    ⇐   split_spec(Spec,X)  ∧  DepthNew = Depth − 1
                                     ∧ decompose_spec(Spec,X,Spec′,Spec″)
                                     ∧ div_and_con(Spec′,Proof′,DepthNew)
                                     ∧ div_and_con(Spec″,Proof″,DepthNew)
                                     ∧ compose_proof(Proof′,Proof″,Proof).
```

Applying generic methods to prove goals instead of general purpose calculi may reduce the computational complexity, result in terminating procedures or make proofs simpler. Generic methods are derived once and can be applied to prove a class of goals, rather than simply single instances. They offer the choice to employ strategic knowledge such as behavioural equivalences or solved cases in an intelligible way for the control of proof processes. Proof Terms generated by metalevel programs may provide useful explanations, since - in difference to complete proof terms on the object level - only major abstract proof steps are presented.

4.3 INTERFACES

The metalevel describes valid proof steps on the domain level and relations among the derivability of specifications. To execute concrete computations, the domain knowledge has to be linked to the metalevel.

Each individual application specific binding of generic domain theories is called *interpretation.* The inverse process - mapping domain theories onto generic domain theories - is called *abstraction.* Abstractions structure concrete domain theories in terms of generic expressions. Abstractions and Interpretations are called *Interface Declarations.*

An *Interface Declaration* consists of the definition of *lifting axioms*, and *implementations* for the parametric metapredicates. Lifting axioms describe well defined object level substitutions for the corresponding terms on the metalevel, ie. constants, function and predicate symbols, atoms and formulas. Correct implementations for the external metapredicates are to be provided with respect to the stated soundness axioms.

By the following axioms the topological sorting theory is lifted into a decomposition theory.

```
Interface                                  DecomposeTopSort
Imports                                    TopSort,Formula

lifting                                    spec → bool

Lifting
lifting(∀U[∃Z : iso(U,P) ⇒ topsort(U,Z)]).

Basic Axioms
minimal_spec(Spec)                    ⇐   destruct(pre_arg,Spec,P)  ∧  totally_ordered(P).
split_spec(Spec,X)                    ⇐   destruct(pre_arg,Spec,P)  ∧  new_rel(P,X).
decompose_spec(Spec,X,Spec′,Spec″)    ⇐   destruct(pre_arg,Spec,P)
                                           ∧ all_case_solutions(reduct(P,X),[P′,P"])
                                           ∧ replace(pre_arg,P′,Spec,Spec′)
                                           ∧ replace(pre_arg,P″,Spec,Spec″).

Strategic Knowledge
behav_equiv(Spec,SpecEquiv)           ⇐   destruct(pre_arg,Spec,P)
                                           ∧ destruct(pre_arg,SpecEquiv,PEquiv)
                                           ∧ iso(P,PEquiv).
```

Hence, proving the goal

$$\Leftarrow \mathtt{div\_and\_con}(\Leftarrow \forall \mathtt{P}[\exists \mathtt{S} : \mathtt{iso}(\mathtt{P},[\{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e}\},\{\}]) \Rightarrow \mathtt{topsort}(\mathtt{P},\mathtt{S})],\mathtt{Proof},7)$$

causes the the variable Proof to be instantiated with an abstract description of the optimal algorithm (cf. [12]), which sorts an arbitrary set of 5 elements with at most 7 comparisons.

Especially, this example reveals that metalevel reasoning involves more than only homomorphism, where well defined entities are assigned new names. A major aspect of domain types is its enriched expressiveness. Employing domain types, additional knowledge as solved cases can be provided optionally. In the above example, an axiom which concerns behavioural equivalence is stated.

5 KNOWLEDGE STRUCTURING AND TRANSFORMATION

Knowledge is subject to continual changes. Thus, a knowledge based program development system ought support static and dynamic views on knowledge. Easy addition, deletion and refinement of knowledge sources have to be facilitated to implement knowledge acquisition, program construction, program verification or program transformation methods.

Logic programming languages like Prolog [4] are based on the premise that computation is done from a static unique theory. The unique theory assumption may result in name clashes or in inconsistencies, if several distinct domain theories have to be considered. The dynamic change of a program can be implemented in Prolog by `assert` and `retract` or by using non-standard extensions as an interface to the system environment. This is unsatisfying, because, as some side effects may occur, even clearly structured abstract specifications cannot be implemented in a straightforward way. Instead, they have to be programmed by use of tricky idiosyncrasies of the given environment. The resulting programs scarcely reflect the logical structure of the original specification.

In this section, our general approach - distinguishing data types and declarative specifications - is extended, thus providing a logic-based solution to the mentioned tasks in knowledge representation. A knowledge-based system is conceived as a collection of knowledge sources, which can be organized as an inheritance hierarchy and classified by data types. Knowledge sources are input to various typed logic programs implementing program development methods. Sets of knowledge sources can be bound to variables by context-switching and passed as arguments via their names.

5.1 THEORY TYPES

In general, a semantic universe is considered as a set of individual objects together with properties, relationships among them and operations defined for this domain. The structure and behaviour of objects belonging to the domain are defined by the associated data type specifications.

In the following we adopt this formal approach in order to structure comprehensive knowledge bases. Each theory is treated as an object, which is identified by a logical name, the *theory name* and contains a chunk of knowledge called *value* of the theory object. The value of a *primitive theory* is the set of predicate definitions which are associated with the corresponding theory name. Collections of theories are treated as universes and classes of theories are represented by abstract types on which one can perform different kinds of operations. Thus - as before - a distinction between structured collections of theories, called *theory types*, and *type instances*, ie. individual theories, is made.

A *theory type* is defined by a signature; constraints and axioms. Collections of Theories are classified by sorts due to syntactical or semantical properties of their entailed formulas, eg. horn clause formulas or positive variable free and negative existence quantified formulas, or due to their persistence. Structural restrictions may be axiomatized by *constraints*. Fundamental backtrackable operations, eg. `union`, for interaction and manipulation of arbitrary theories are provided.

Type Construction by Component Terms In the previous sections, we mentioned two construction principles for building new types from already defined ones: the inductive construction of terms and the construction of formulas by composing implication symbols, pre-, and postconditions. As a major principle for the construction of theory types from already defined types, we suggest to employ frame-like data structures. They allow to model *aggregation* relations, such as an "instance of a type `t` is composed of instances of types $\mathtt{t}_1, \mathtt{t}_2, \ldots, \mathtt{t}_r$".

The *signature* of aggregated types is described by specific feature terms (cf. [1, 21]), which we call component terms. A *component term* is a tree, consisting of a type entry, any number (≥ 0) of features, which are assignments of the form `value : component`, where `value` is a theoryname or a variable and `component` is a component term or a sort.

The *semantics* of component terms is as follows: every ground term is assigned a single theory, which is the union of theories associated with terminal nodes; non-ground terms describe classes of theories, whose subcomponents are characterized by common properties. Structural dependencies among components can be stated by *constraints*. For example, the component term `theoryf(A : atoms(P : pos_a, N : neg_a), R : rules, C_prim : formulas)` may be used to structure collections of theories, whose parts are atoms which are subdivided into positive atoms and negative atoms, horn clauses, and construction primitives, respectively, taking for granted that the basic sorts are defined. The component term `dec(..., B : d_bound(X : theoryf))` indicates that the subcomponent `d_bound`, which entails derived bounds of a decomposition theory `dec`, may be structured by the type `theoryf`. Concrete theories are represented in terms of specific instances of theory types, eg. the assignment `sort : dec` declares the sorting theory to be an instance of a decomposition theory.

Operations on aggregated types can be defined in terms of the above mentioned operations on basic theory types. To destruct or construct individual components of aggregated theories, path expressions are used. Path expressions are syntactic constructs allowing to access subterms by names instead of accessing them in a term by their positions. Coreference constraints among paths may be used to refer to identical subcomponents.

Context Switching In the SEAMLESS approach as well as in MetaProlog (cf. [3]), the program synthesis system always runs in a certain context, called *current context* or *background theory*, and all goals are proven with respect to this context. A context represents the currently activated theories by their names. To prove a goal with respect to a certain theory, the predicate `prove` can be used. Invoking `prove(Context,Goal)` causes the axioms associated with the theoryname `Context` to be added on top of the current Context `Current_Context` and the `Goal` to be proven with respect to `Context` ∪ `Current_Context`. If the proof of `Goal` is accomplished, the formulas associated with the name `Context` are removed.

Context switching provides the means for the dynamic assignment of values to theory variables and can be regarded as a counterpart to unification.

5.2 KNOWLEDGE THEORIES

To allow reasoning about classes of theories, we introduce knowledge theories. A *Knowledge Theory* is defined by a typed higher order logic, which consists of theory types, a signature and parametric method descriptions. Parametric methods take sorted theories as arguments. The intended semantics is stated by soundness axioms. For instance[2]:

$\mathtt{subsumption}_{\mathtt{ctx}}(\mathtt{TName},\mathtt{Formula})$ is intended to be true, iff `Formula` is subsumed by the Theory associated with `TName`.

$\mathtt{redundant}_{\mathtt{ctx}}(\mathtt{TName},\mathtt{Formula})$ eliminates in a given theory with the name `TName` all clauses which are (θ)-subsumed by `Formula` and adds `Formula`.

$\mathtt{construct}_{\mathtt{ctx}}(\mathtt{TName},\mathtt{C},\mathtt{D})$ constructs the most general formula `D` due to the construction primitives, specified in `TName`, which subsumes `C`.

$\mathtt{inconsistent}_{\mathtt{ctx}}(\mathtt{TName},\mathtt{C})$ eliminates in a given theory with the name `TName` all clauses which cause $\mathtt{TName} \cup \mathtt{C}$ to be inconsistent.

General purpose procedures which implement these methods - involving arbitrary first order theories - do not exist or have exponential complexities. However, for restricted types of formulas, decidable and efficient procedures have been devised in the literature. Our diversified approach - sorted method descriptions - allows to implement these procedures in a uniform framework by following naming conventions and stating restrictions by constraints.

5.3 KNOWLEDGE TRANSFORMATION METHODS

Knowledge Theories provide the theoretical and implementational basis to devise correct *Generic Knowledge Transformation Method* in terms of the basic settheoretic operations and parametric methods. This will be exemplified by considering theory formation tasks. Generally, the theory formation task can be stated as follows:

Given a set of atoms - labelled as positive or negative - $\mathtt{ex}_1, \mathtt{ex}_2, \ldots = \mathtt{ex}^+ \cup \mathtt{ex}^-$ and a consistent background theory named `background_theory`, which may be empty, construct a minimal set of formulas named `c_theory`, such that

$\forall \mathtt{Ex} \in \mathtt{ex}^+ : \mathtt{subsumption}(\mathtt{c\_theory} \cup \mathtt{background\_theory}, \mathtt{Ex}).$
$\forall \mathtt{Ex} \in \mathtt{ex}^- : \mathtt{consistent}(\mathtt{c\_theory} \cup \mathtt{Ex}).$

[2] It has to be considered that theories are accessed by their names. That means, although theory values are being transformed by applying operations, names are not necessarily subject to change. Context-changing operations are indexed by `ctx`

The following axioms sketch a constructive generic method for theory formation. However, in this paper, we only require the constructed theory to cover a maximal subset of the positive examples in a consistent way.

Knowledge Transformation Method		Theory − Formation
Imports		Interface(−)
Predicates		
theory_form$_{ctx}$:		theoryf → bool
debug$_{ctx}$:		theoryf × formula → bool
abstract_integrate$_{ctx}$:		theoryf × atom → bool
Axioms		
theory_form$_{ctx}$(Context)	⇐	destruct(Context,atoms,Exs) ∧ forall(Ex,Exs,abstract_integrate$_{ctx}$(Context,Ex)) ∧ destruct(Context,rule,C_Theory) ∧ display(C_Theory).
abstract_integrate$_{ctx}$(Context,Fact)	⇐	construct$_{ctx}$(Context,Fact,Rules) ∧ debug$_{ctx}$(Context,Rules).
debug$_{ctx}$(Context,Rules)	⇐	subsumption$_{ctx}$(Context,Rules).
debug$_{ctx}$(Context,Rules)	⇐	inconsistent$_{ctx}$(Context,Rules).
debug$_{ctx}$(Context,Rules)	⇐	redundant$_{ctx}$(Context,Rules).

This generic method may be applied for the solution of concrete theory formation tasks. Proving the goal

⇐ prove(i_t_b, theoryform(sort.d_bound))

causes the parametric import declaration to be bound by the interface i_t_b, which provides correct implementations for the parametric methods as subsumption; a theory sort.d_bound.rules to be constructed and displayed, which entails the general upper and lower bounds to topological sorting problems, inductively derived from specific instances by applying the method theoryform.

6 CONCLUSIONS

This paper has clarified how to model several forms of domain and program synthesis knowledge, how to derive correct, constructive methods, and how to state program synthesis tasks within a coherent logical framework with a clean declarative semantics. A three-layer abstraction hierarchy, each layer subdivided into a data type and logic level, is required. The introduction of data-types allows to express control and dynamic aspects explicitly. Especially knowledge bases are treated as abstract data types on which one can perform different operations. The interaction among layers is an object-metalevel relation, implemented by well-defined interfaces. Using a descriptive logic, goals on different levels may be entered as queries and solved constructively by theorem proving.

The SEAMLESS framework suggests to develop knowledge based program synthesis systems as libraries of encapsulated knowledge entities, ie. data types, domain descriptions and a workbench of inference-based tools, ie. generic methods for constructing specific purpose systems. The approach enables the re-usability of knowledge entities and allows to analyze problems from multiple views or levels of abstractions. Hence, evolutionary program synthesis is facilitated.

Other proposals have been made to formalize problem solving knowledge within a higher order approach. In [13] the basic principles of a very general theory of program construction on top of Intuitionistic Type Theory are presented. It allows to formulate both object and meta-knowledge within some unified framework. It is revealed that correct and semantically well understood program synthesis methods can be derived by proving metatheorems. The distinction of different layers is also a central feature of the KADS methodology, which is devised to model problem solving expertise. KADS models consist of four layers: domain, inference, task and strategy. $(ML)^2$ [23] and VITAL [11] are formalizations of KADS conceptual models. In both approaches - as in our approach - order sorted predicate logic is used to specify the domain knowledge. None of the above cited approaches presents a formalization of the strategic layer. A major deficiency of recent formalizations of KADS conceptual models is that the relations between the different layers and the semantics of the "lifting" operations remain unclear. The mapping from domain knowledge to parametric schemas and the definitions of problem solving methods are defined in an ad-hoc way.

We implemented a prototypical system ASK ("Acquiring Synthesis Knowledge"), which presently supports a subset of the envisaged system. In ASK, methods for knowledge acquisition by learning from examples, the application of a program synthesis tactic and verification by inductive proofs are integrated. It was used to assist the discovery of a previous unknown, complex mathematical algorithm (cf. [6, 7]). Search complexities were tackled by extended strategic knowledge, which was implemented as annotated expert knowledge and automatically augmented at runtime by machine learning procedures. Currently, we continue to build a configuration toolkit for the construction of individualized telecooperation systems on top of our ASK-System and the SEAMLESS approach. Following the proposed framework, major tasks are to locate adequate generic goal structures and generic methods and to make configuration knowledge accessible.

References

[1] H. Ait-Kaci and R. Nasr. Login, a logic programming language with built-in inheritance. *Journal of Logic Programming*, 3:185–215, 1986.

[2] A. Borgida and P. Devanbu, Premkumar. Knowledge base management systems using description logics and their role in software information systems. In F.H. Vogt, editor, *Personal Computers and Intelligent Systems, Information Processing 92*, volume 3, pages 171–181. Elsevier Science Publishers, 1992.

[3] I. Cicekli. Design and implementation of an abstract metaprolog engine for metaprolog. In H. Abramson and M. Rogers, editors, *Meta-Programming in Logic Programming*, pages 417–434. MIT Press, 1989.

[4] W.F. Clocksin and C.S. Mellish. *Programming in PROLOG*. Springer Verlag, 1981.

[5] S.L. Epstein and N.S. Sridharan. Knowledge representation for mathematical discovery: Three experiments in graph theory. *J. of Applied Intelligence*, 1(1):7–32, 1991.

[6] J. Eusterbrock. Errata to "Selecting the top three elements" by M. Aigner: A Result of a computer assisted proof search. *Discrete Applied Mathematics*, 41:131–137, 1992.

[7] J. Eusterbrock. *Wissensbasierte Verfahren zur Synthese mathematischer Beweise: Eine kombinatorische Anwendung*, volume 10 of *DISKI*. INFIX, 1992.

[8] C. Floyd. A systematic look at prototyping. In R. Budde, editor, *Approaches to Prototyping*. Springer, 1984.

[9] J.A. Goguen and J. Meseguer. Unifying functional, object-oriented and relational programming with logical semantics. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 417–477. MIT Press, 1987.

[10] P.M. Hill and J.W. Lloyd. The Gödel report. Technical Report TR-91-02, University of Bristol, 1991.

[11] W. Jonker and J.W. Spee. Yet another formalisation of KADS Conceptual Models. In *Current Developments in Knowledge Acquistion - EKAW92*, pages 211–229. Springer Verlag, 1992.

[12] D.E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison Wesley, Reading, MA, 1973.

[13] Chr. Kreitz. The representation of program synthesis in higher order logic. In H. Marburger, editor, *GWAI-90, 14th German Workshop on Artificial Intelligence*, pages 171–180. Springer Verlag, 1990.

[14] C.W.H. Lam, L.H. Thiel, and S. Swiercz. A computer search for a projective plane of order 10. In M.M. Deza, P. Frankl, and I.G. Rosenberg, editors, *Algebraic, Extremal and Metric Combinatorics*. London Math. Soc. Lecture Note Series 131, 1988.

[15] M.R. Lowry. *Algorithm Synthesis through Problem Reformulation*. PhD thesis, Stanford University, 1989.

[16] M.R. Lowry. Methodologies for knowledge-based software engineering, Proc. 7th international symposium, ISMIS '93. In J. Komorowski and Z.W. Ràs, editors, *Methodologies for Intelligent Systems*, pages 219–234. Springer Verlag, 1993.

[17] S. Ohsuga. How can knowledge-based systems solve large-scale problems: model based decomposition and problem solving. *Knowledge-Based Systems*, 6(1):38–62, 1993.

[18] I. Parberry. A computer-assisted optimal depth lower bound for nine-input sorting networks. *Mathematical Systems Theory*, 24:101–116, 1991.

[19] E.Y. Shapiro. *Algorithmic Program Debugging*. ACM Distinguished Dissertations, The MIT Press, 1982.

[20] D.R. Smith. Automating the design of algorithms. In St.S̃chuman B.M̃öller, H.P̃artsch, editor, *Formal Program Development*, pages 324–354. Springer Verlag, 1992.

[21] G. Smolka and R. Treinen. Records for logic programming. *Journal Logic Programming*, 18:229–258, 1994.

[22] J.P. Tsai, Th. Weigert, and H.C. Jang. A hybrid knowledge representation as a basis of requirement specification and specification analysis. *IEEE Transactions on Software Engineering*, 18(12):1076–1100, 1992.

[23] F. van Harmelen and J. Balder. $(ML)^2$: A formal language for KADS models. In *Proc. of the 10th European Conference on Artificial Intelligence*, pages 582–586. John Wiley & Sons, 1992.

[24] G. Wiggins, A. Bundy, I. Kraan, and J. Hesketh. Synthesis and transformation of logic programs by constructive inductive proof. In K-K. Lau und T. Clement, editor, *Logic Program Synthesis and Transformation*, 1992.

INTEGRATION OF RULE INFERENCES INTO A TERMINOLOGICAL COMPONENT

Peter Forster and Bernd Novotny

Institut für Informatik, Universität Stuttgart
Breitwiesenstr. 20-22, D-70565, Germany
email: {forster, novotny}@informatik.uni-stuttgart.de

ABSTRACT

In this paper, we address the problem of integrating rule inferences into a terminological representation system. The rule component allows the use of n-ary relationships by introducing arbitrary n-ary predicates. In order to get a well-defined semantics, we restrict the form of our rules to a special kind of Horn clauses. Our approach of integrating rules into a terminological representation system leads to inconsistencies. We propose a solution which allows to detect some of the inconsistencies.

1 INTRODUCTION

Knowledge representation systems based on the idea of KL-ONE, so-called terminological representation systems [1], have led to the development of many systems during the last years. These systems can be used to describe concepts of an arbitrary domain and the relationships between these concepts. Additionally, many systems provide a component to state and retrieve assertions about individuals w.r.t. a particular domain, e.g. "`Peter is a human`" is an assertion, meaning that the individual `peter` is an instance of the concept `human`. Terminological representation systems have been used in a variety of applications, such as configuration systems [2] and natural language processing systems [3]. In order to be useful in *real world* applications, it turns out that the expressive power of the formalisms has to be increased.

In standard terminological representation systems relationships between concepts are restricted to binary relations. On the other hand there is often a need in applications to represent arbitrary n-ary relations, for example to represent spatial relationships like `between(x,y,z)`, meaning that an object `x` is located between the objects `y` and `z`. Another drawback is that the inferences do not consider contextual knowledge. For example, it is not possible to declare "partial transitive" relations.

Partial transitive relations play an important role in the field of computational linguistics. Consider the merological relations `(part-of door house)` and `(part-of handle door)` [4]. In this context, it is obvious that `part-of` is not transitive. In the context `(part-of exhaust car)`, `(part-of catalyst exhaust)` the transitive conclusion `(part-of catalyst car)` seems to be valid.

In this paper, we propose to integrate this kind of knowledge in form of rules into a terminological formalism. In our approach, the rule component derives additional knowledge on demand, i.e. the rule component does not complete the assertional knowledge permanently. Inferences drawn from the terminological or assertional part of the knowledge representation system have a priority to inferences drawn from the rule component. In order to be able to define a well-defined semantics for the rule component the expressiveness of the rule component is restricted.

This paper is organized as follows. First we recall the basic terms of terminological representation systems. Then, we describe syntax and semantics of a terminological formalism, used as platform for the rule integration. In the next section, we describe our approach more formally and discuss problems caused by this approach. Finally, we give an outlook to related work.

2 TERMINOLOGICAL REPRESENTATION SYSTEMS

Terminological representation systems are based on terminological logic [5]. The essential objects are structured terms, called *concepts* which are related to each other by binary relationships, called *roles*. Concepts are used to describe the knowledge base (KB) schema of the representation system. They are organized in a taxonomy. Concepts are defined with respect to more general concepts and by restricting roles. Particularly, roles can be restricted by the number (*number-restrictions*) and the range (*type-restrictions*) of allowed role fillers. The most important inference operating on the KB schema is the *classification* which inserts new concepts in a taxonomy at the correct place. The classification process is supported by the subsumption relation, determining whether a concept $C1$ is more general than another concept $C2$, written $C1 \sqsupseteq C2$.

In addition, most of the terminological representation systems provide an assertional component, the ABox. This component enables to state facts about *individuals* of the form: an individual o is an instance of a concept C, written $(C\ o)$, and two individuals o, p are connected by a role R, written $(R\ o\ p)$. The most important inference operating on the assertional component is the *realization* or individual classification [6] by which all concepts C_i of a given terminology satisfied by an individual object o are determined, written $o \in C_i$. Typical retrieval operations for the assertional component are tests whether given hypotheses are valid with respect to a given ABox $\mathcal{W}_\mathcal{A}$, written $\mathcal{W}_\mathcal{A} \models (C\ o)$ and $\mathcal{W}_\mathcal{A} \models (R\ o\ p)$, respectively. Terminological representation systems usually have a well-defined semantics for their representational constructs.

Definition 2.1 (Syntax of the terminological representation language) *The domain $\mathcal{D}$ is an arbitrary set. The terminological language we will use is defined as follows:*

$$\begin{array}{lcll} C, D & \rightarrow & \top \mid & \text{(top)} \\ & & \bot \mid & \text{(bottom)} \\ & & A \mid & \text{(atomic concept)} \\ & & (C \sqcap D) \mid & \text{(concept conjunction)} \\ & & (\text{all } R\ C) \mid & \text{(value restriction)} \\ & & (\text{atmost } R\ n) \mid & \text{(atmost restriction)} \\ & & (\text{atleast } R\ n) & \text{(atleast restriction)} \\ R & \rightarrow & P & \text{(atomic role)} \end{array}$$

Here A denotes an atomic concept, C and D arbitrary concepts, R an arbitrary role and n a positive integer. Terminological axioms are used to introduce names for concepts. Let A be a concept name and D be a concept then we call the following forms terminological axioms:

$A \doteq D$ *to introduce primitive concepts (the conditions D for A are necessary).*

$A \dot{\sqsubseteq} D$ *to introduce defined concepts (the conditions D for A are necessary and sufficient).*

A finite set of terminological axioms is called a terminology *or TBox $\mathcal{T}$ with the additional restrictions:*

(i) every concept name used in $\mathcal{T}$ must appear exactly once on the left hand side of a terminological axiom.

(ii) $\mathcal{T}$ must not contain cyclic definitions.

$\mathcal{C}$ and $\mathcal{R}$ are sets of concept names and role names, respectively.

The semantics of a TBox $\mathcal{T}$ is given by defining interpretations and models.

Definition 2.2 (Semantics of the terminological representation language) *An interpretation $\mathcal{I}(\mathcal{D}, \cdot^{\mathcal{I}})$ of our language consists of a set $\mathcal{D}$, the domain of $\mathcal{I}$ and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function is mapping concepts to subsets of $\mathcal{D}$ and roles to subsets of $\mathcal{D} \times \mathcal{D}$, such that the following equations are satisfied:*

$$\begin{array}{lcl} \top^{\mathcal{I}} & = & D^{\mathcal{I}} \\ \bot^{\mathcal{I}} & = & \emptyset \\ (C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\text{all } R\ C)^{\mathcal{I}} & = & \{x \in \mathcal{D} \mid \forall \langle x, y \rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\} \\ (\text{atmost } R\ n)^{\mathcal{I}} & = & \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in R^{\mathcal{I}}\}\| \leq n\} \\ (\text{atleast } R\ n)^{\mathcal{I}} & = & \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in R^{\mathcal{I}}\}\| \geq n\} \end{array}$$

An interpretation $\mathcal{I}$ satisfies a terminological axiom $A \doteq D$ or $A \dot{\sqsubseteq} D$ iff $A^{\mathcal{I}} = D^{\mathcal{I}}$ respectively $A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies all terminological axioms in $\mathcal{T}$.

The *subsumption* relation organizes the concepts of a TBox. The computation of the subsumption hierarchy, i.e., computing the subconcept-superconcept relations between the concepts of a TBox is usually called *concept classification*. The semantics introduced above allows the formal definition of subsumption.

Definition 2.3 (Subsumption) *Let $\mathcal{T}$ be a Tbox and C and D concepts. Then D subsumes C, written $C \sqsubseteq D$, with respect to $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$.*

The assertional component supplements the TBox. It enables to state facts about individuals and to restrict relationships between individuals.

Definition 2.4 (Syntax and Semantics of an ABox) *Let $\mathcal{O}$ be an alphabet of symbols, called individuals and o, p be individuals. Then $(C\ o)$(meaning that o is an instance of C) and $(R\ o\ p)$(meaninig o is in relation to p by means of R) are assertional axioms. $\mathcal{W}_{\mathcal{A}}$ is a finite set of assertional axioms and is called a* world description *or ABox. The interpretation of a world description $\mathcal{W}_{\mathcal{A}}$ is an interpretation $\mathcal{I}$ with the extension that the interpretation function $\cdot^{\mathcal{I}}$ assigns an element $o^{\mathcal{I}} \in \mathcal{D}$ to each individual o. An interpretation $\mathcal{I}$ satisfies an assertional axiom δ:*

$$\begin{aligned} \delta &= (C\ o) & \textit{iff}\ \ o^{\mathcal{I}} \in C^{\mathcal{I}} \\ \delta &= (R\ o\ p) & \textit{iff}\ \ \langle o^{\mathcal{I}}, p^{\mathcal{I}} \rangle \in R^{\mathcal{I}} \end{aligned}$$

An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{W}_{\mathcal{A}}$ iff it satisfies all $\delta \in \mathcal{W}_{\mathcal{A}}$. An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{W}_{\mathcal{A}}$ w.r.t. a TBox $\mathcal{T}$, iff $\mathcal{I}$ is a model of $\mathcal{W}_{\mathcal{A}}$ and $\mathcal{T}$. An ABox $\mathcal{W}_{\mathcal{A}}$ and a TBox $\mathcal{T}$ imply a description δ, written $\mathcal{W}_{\mathcal{A}} \models_{\mathcal{T}} \delta$, iff all models of $\mathcal{W}_{\mathcal{A}}$ w.r.t. $\mathcal{T}$ satisfy δ.

3 THE INTEGRATION OF RULES

3.1 MOTIVATION

In our approach, the basic idea of integrating rules into a terminological representation system is that rules derive *additional knowledge* on *demand*. *Additional knowledge* means relationships between instances or between instances and concepts, that are not stored explicitely in the ABox. Relationships infered from the rule component differ from the inference capabilities of the terminological formalism. The rule component allows the use of n-ary instance relationships by introducing arbitrary n-ary predicates. In contrast to rule relationships, relationships in ordinary terminological representation systems are restricted to binary relations.

To derive knowledge on *demand* means that the knowledge deduced by the rule component is not stored in the ABox. Thus, using rules reduces the amount of stored relationships in the ABox. Inferences on the rule component are made by *backward* *chaining.* This inference mechanism makes it possible to derive additional knowledge on demand, without changing the ABox. In this solution, *backward chaining* in combination with Horn clauses avoids a revision of the ABox.

A basic requirement for the integration of rules into a terminological representation system is the existence of a well-defined semantics of the rule-based inferences. This is necessary to understand the behaviour of the whole system. Therefore, we restrict our rules to expressions based on first order logic and do not combine procedural attachment features into the rule approach.

3.2 RULE INTEGRATION

3.2.1 Rules

To guarantee a well-defined semantics of the rule inferences, we make use of the similarity between rules and logical implications. Using logical implications as rules enables to define the semantics of the rule inferences by means of the *predicate calculus.* We restrict our rules to Horn clauses:

$$A \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_m$$

where $A, B_1, \ldots, B_m$ are arbitrary n-ary predicates. These rules are sufficient for describing additional knowledge for our applications, i.e. relationships between instances and between instances and concepts.

3.2.2 Semantics of the Rule Inferences

The rules defined by the user establish a set of Horn clauses, a *Horn clause program.* Some of the predicates used here may correspond to concept names or role names defined in the TBox. Concept names are considered as symbols of unary predicates, role names as symbols of binary predicates.

Rules: User-defined rules in Horn clause form represent the additional knowledge.

Rules establish only new relationships between individuals, respectively concepts, and must not introduce additional individuals. Therefore, we restrict the permitted rules to *range-restricted function-free definite implicational clauses.* A *definite implicational clause* has at least one literal in the body and exactly one literal in the head. *Range-restricted* means that the variables in the head are a subset of the variables in the body [7].

Moreover, we prohibit the use of *constants* with no corresponding individual in the ABox.

The knowledge from the terminological representation system, the subsumption relations and the facts can be integrated in the following way:

Subsumption Relations: Knowledge about a subsumption relation can be described as a rule in Horn clause form:

$$Concepts:\ C_1(x) \leftarrow C_2(x).\quad if\ \ C_1^{\mathcal{I}} \sqsupseteq C_2^{\mathcal{I}}$$

Here C_1 and C_2 denote concept names.

Facts: An assertional statement $(C\ o)$ is transformed into a unit clause consisting of the unary predicate $C(o)$ if C denotes a concept name. An assertional statement $(R\ o\ p)$ is transformed into a unit clause consisting of the binary predicate $R(o,p)$.
There are no assertional statements in the ABox corresponding to n-ary predicates with n>2.

The knowledge needed for the rule inferences is transformed into a Horn clause program. This transformation step can be formally defined as follows.

Definition 3.1 (Horn Clause Program $\mathcal{HC}$) *Let $\mathcal{T}$ be a TBox, $\mathcal{W}_{\mathcal{A}}$ be an ABox, $\mathcal{C}$ be the set of concept names of $\mathcal{T}$ and $\mathcal{R}$ the set of role names of $\mathcal{T}$.*
Let $\mathcal{HC}$ be a set of Horn clauses, $\mathcal{HC} = \mathcal{H}_{\mathcal{R}} \cup \mathcal{H}_{\mathcal{T}} \cup \mathcal{H}_{\mathcal{W}}$, with:

- $\mathcal{H}_{\mathcal{R}}$ *= a set of range-restricted function-free definite implicational clauses established by the user-defined rules.*
- $\mathcal{H}_{\mathcal{T}} := \{C_1(x) \leftarrow C_2(x) \mid C_1, C_2 \in \mathcal{C}\ \wedge\ C_1^{\mathcal{I}} \sqsupseteq C_2^{\mathcal{I}}\}$
  (subsumption relations)
- $\mathcal{H}_{\mathcal{W}} := \{C(o) \leftarrow\ \mid o^{\mathcal{I}} \in C^{\mathcal{I}} \wedge C \in \mathcal{C}\}\ \cup\ \{R(o,p) \leftarrow\ \mid \langle o^{\mathcal{I}}, p^{\mathcal{I}}\rangle \in R^{\mathcal{I}} \wedge R \in \mathcal{R}\}$
  (facts)

The range-restrictedness of the clauses in $\mathcal{HC}$ allows to define a finite model of $\mathcal{HC}$. In the description of the semantics of the extended formalism, we will refer to this model.

Definition 3.2 (Atomic-derivation-closure) *The n-atomic-derivation-set of $\mathcal{HC}$, $D^n(\mathcal{HC})$ is defined recursively as follows.*
$D^0(\mathcal{HC})$ is the set of ground unit clauses in $\mathcal{HC}$: $D^0(\mathcal{HC}) = \mathcal{H}_{\mathcal{W}}$
$D^n(\mathcal{HC}) = D^{n-1}(\mathcal{HC}) \cup \{A\theta_1 \ldots \theta_m \mid A \leftarrow B_1, \ldots, B_m \in \mathcal{HC}$ and for each B_i there exists $B_i' \in D^{n-1}(\mathcal{HC})$ such that θ_i is the most general unifier of B_i and $B_i'\}$.
The atomic-derivation-closure $D^\star(\mathcal{HC})$ is the set $D^0(\mathcal{HC}) \cup D^1(\mathcal{HC}) \ldots$

The definition of $\mathcal{HC}$ (definition 3.1) and the definition of $D^\star$ (definition 3.2) imply, that $D^\star(\mathcal{HC})$ is finite (proof [8]) and every element of $D^\star(\mathcal{HC})$ is ground (proof [8]). To infer n-ary relations in the extended formalism, the set of assertional axioms is supplemented.

Definition 3.3 (Universal Axiom) *An axiom Ω is a universal axiom, if:*

- *Ω is an assertional axiom, or*

- *Ω is an arbitrary term of the form $(P\ o_1 \ldots o_n)$, with $n \geq 1$.*

This leads to the definition of the semantics of the world description established by the Horn clause program $\mathcal{HC}$. To define a semantics for a universal axiom $((atmost\ R\ n)\ o)$ does not seem to be appropriate, since $D^\star(\mathcal{HC})$ contains only the current relationships between instances and does not give any information about the maximum of possible relationships.

Definition 3.4 (Semantics of the Rule-world Description) *Let $\mathcal{HC}$ be a Horn clause program, $\mathcal{W}_\mathcal{A}$ be an ABox. $\mathcal{W}_{HC}$ is a finite set of universal axioms, called* rule-world description.
An interpretation $\mathcal{I}_{HC}(\{true, false\}, \cdot^{\mathcal{I}_{HC}})$ of $\mathcal{W}_{HC}$ consists of a set $\{true, false\}$, the domain of $\mathcal{I}_{HC}$ and an interpretation function $\cdot^{\mathcal{I}_{HC}}$. The interpretation function is mapping universal axioms to an element of $\{true, false\}$:

$$\begin{aligned}
(P\ o_1 \ldots o_n)^{\mathcal{I}_{HC}} &= P(o_1, \ldots, o_n) \in D^\star(\mathcal{HC}) \\
((C \sqcap D)\ o)^{\mathcal{I}_{HC}} &= (C\ o)^{\mathcal{I}_{HC}} \wedge (D\ o)^{\mathcal{I}_{HC}} \\
((atleast\ R\ n)\ o)^{\mathcal{I}_{HC}} &= \|\{p \mid R(o,p) \in D^\star(\mathcal{HC})\ \}\| \geq n \\
((all\ R\ C)\ o)^{\mathcal{I}_{HC}} &= \exists n :\ \mathcal{W} \models_\mathcal{T} ((atmost\ R\ n)\ o) \\
&\qquad\quad \wedge \|\{p \mid R(o,p) \in D^\star(\mathcal{HC})\ \}\| = n \\
&\qquad\quad \wedge (\forall p_i \in \{p \mid R(o,p) \in D^\star(\mathcal{HC})\ \} : \\
&\qquad\qquad\quad \mathcal{W} \models_\mathcal{T} (C\ p_i)\ ) \\
((atmost\ R\ n)\ o)^{\mathcal{I}_{HC}} &= false
\end{aligned}$$

An interpretation $\mathcal{I}_{HC}$ satisfies a universal axiom Ω iff $\Omega^{\mathcal{I}_{HC}} = true$.
An interpretation $\mathcal{I}_{HC}$ is a model of $\mathcal{W}_{HC}$ iff it satisfies all universal axioms $\Omega \in \mathcal{W}_{HC}$.

3.2.3 The Extended Formalism

After defining when an ABox $\mathcal{W}_\mathcal{A}$ satisfies an assertional axiom δ ($\mathcal{W}_\mathcal{A} \models_\mathcal{T} \delta$), and when a rule-world description $\mathcal{W}_{HC}$ satisfies a universal axiom Ω ($\mathcal{W}_{HC} \models \Omega$), the next step is to integrate both descriptions, $\mathcal{W}_\mathcal{A}$ and $\mathcal{W}_{HC}$. This leads to the extended formalism. The semantics is defined as follows:

Definition 3.5 (Extended Formalism) *Let $\mathcal{W}_{\mathcal{A}+HC}$ be a so-called* universal description, *consisting of an ABox $\mathcal{W}_\mathcal{A}$ and a rule-world description $\mathcal{W}_{HC}$. A universal axiom Ω is valid with respect to $\mathcal{W}_{\mathcal{A}+HC}$, written $\mathcal{W}_{\mathcal{A}+HC} \models \Omega$, if:*

$$\mathcal{W}_\mathcal{A} \models_\mathcal{T} \Omega \ \vee \ \mathcal{W}_{HC} \models \Omega$$

Example 3.1:

Consider the example of spatial relationships mentioned in the introduction. This example illustrates the integration of a 3-ary relationship. ENCIRCLED-OBJECT is dependent on the 3-ary relation BETWEEN.

Given:

The ABox:

$\mathcal{W}_{\mathcal{A}}$: (OBJECT Cube)
(OBJECT Pyramid)
(OBJECT Cylinder)
(RIGHT-OF Cube Pyramid)
(LEFT-OF Cube Cylinder)

The user-defined rules:

$\mathcal{H}_{\mathcal{R}}$: BETWEEN(x,y,z) $\leftarrow$ RIGHT-OF(x,y), LEFT-OF(x,z).
ENCIRCLED-OBJECT(x) $\leftarrow$ OBJECT(x), OBJECT(y), OBJECT(z),
BETWEEN(x,y,z).

$\mathcal{H}_{\mathcal{T}}$ is empty, since there are no subsumption relations in the TBox.

To be shown:

Is the universal axiom Ω = (ENCIRCLED-OBJECT Cube) valid w.r.t. the universal description $\mathcal{W}_{\mathcal{A}+\mathcal{HC}}$?
It is obvious, that $\mathcal{W}_{\mathcal{A}} \not\models_{\mathcal{T}}$ (ENCIRCLED-OBJECT Cube), but it holds that $\mathcal{W}_{\mathcal{HC}} \models$ (ENCIRCLED-OBJECT Cube), because of
ENCIRCLED-OBJECT(Cube) $\in D^{\star}(\mathcal{HC})$.
Therefore: $\mathcal{W}_{\mathcal{A}+\mathcal{HC}} \models$ (Encircled-object Cube)

The other example mentioned in the introduction is about *partial transitive* relations.

Example 3.2:

This example demonstrates the integration of *partial transitive* relations. The user-defined rule restricts the transitivity of the PART-OF relation to CATALYSTS, EXHAUSTS and CARS.

Given:

The ABox:

$\mathcal{W}_{\mathcal{A}}$: (CAR car#1)
(EXHAUST exhaust#1)
(CATALYST catalyst#1)
(PART-OF catalyst#1 exhaust#1)
(PART-OF exhaust#1 car#1)
(HOUSE house#1)
(DOOR door#1)
(HANDLE handle#1)
(PART-OF handle#1 door#1)
(PART-OF door#1 house#1)

The user-defined rules:

$\mathcal{H}_{\mathcal{R}}$: PART-OF(x,y) $\leftarrow$ CATALYST(x), CAR(y), EXHAUST(z),
PART-OF(x,z), PART-OF(z,y).

$\mathcal{H}_{\mathcal{T}} = \emptyset$

The universal axiom Ω_1 = (PART-OF catalyst#1 car#1) is valid w.r.t. $\mathcal{W}_{\mathcal{A}+\mathcal{HC}}$, since:

$\mathcal{W}_{\mathcal{HC}} \models \Omega_1$

but the universal axiom Ω_2 = (PART-OF handle#1 house#1) is not valid w.r.t. $\mathcal{W}_{\mathcal{A}+\mathcal{HC}}$, since:

$\mathcal{W}_{\mathcal{A}} \not\models_{\mathcal{T}} \Omega_2 \;\wedge\; \mathcal{W}_{\mathcal{HC}} \not\models \Omega_2$

3.2.4 Inconsistencies

A consequence of the semantics of the extended formalism is, that the universal description $\mathcal{W}_{\mathcal{A}+HC}$ might be inconsistent, i.e. there might be a pair of universal axioms Ω_1, Ω_2 where Ω_1 is inconsistent with Ω_2.

Definition 3.6 (Inconsistency of Universal Axioms) *Let Ω_1 and Ω_2 be universal axioms. Ω_1 is inconsistent with Ω_2, if one of the following situations holds:*

1. $\Omega_1 = (C_1\ o)\ \wedge\ \Omega_2 = (C_2\ o)\ \wedge\ (C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}) = \emptyset$
2. $\Omega_1 = ((atmost\ R\ n)\ o)\ \wedge\ \Omega_2 = ((atleast\ R\ n+1)\ o)$
3. $\Omega_1 = ((all\ R\ C)\ o)\ \wedge\ \Omega_2 = (R\ o\ p)\ \wedge\ p^{\mathcal{I}} \notin C^{\mathcal{I}}$

Definition 3.7 (Inconsistency of $\mathcal{W}_{\mathcal{A}+HC}$) *Let Ω_1 and Ω_2 be universal axioms, where Ω_1 is inconsistent with Ω_2. A universal description $\mathcal{W}_{\mathcal{A}+HC}$ is inconsistent, if:*

$$\mathcal{W}_{\mathcal{A}+HC} \models \Omega_1 \quad \wedge \quad \mathcal{W}_{\mathcal{A}+HC} \models \Omega_2$$

Example 3.3:
This example illustrates a situation where the universal description $\mathcal{W}_{\mathcal{A}+HC}$ is inconsistent.

Given:

$\mathcal{W}_{\mathcal{A}}$: ((atmost LEFT-OF 0) Cube) (RIGHT-OF Pyramid Cube)

$\mathcal{H}_{\mathcal{R}}$: LEFT-OF(x,y) ← RIGHT-OF(y,x).

Consider the universal axioms:

Ω_1 = ((atmost LEFT-OF 0) Cube)

Ω_2 = ((atleast LEFT-OF 1) Cube)

Ω_1 is inconsistent with Ω_2, but $\mathcal{W}_{\mathcal{A}+HC}$ satisfies both universal axioms:

1. $\mathcal{W}_{\mathcal{A}+HC} \models \Omega_1$, since $\mathcal{W}_{\mathcal{A}} \models_{\mathcal{T}} \Omega_1$
2. $\mathcal{W}_{\mathcal{A}+HC} \models \Omega_2$, since $\mathcal{W}_{\mathcal{H}C} \models \Omega_2$

The occurrence of inconsistencies is a consequence of the semantics of the rule-world description $\mathcal{W}_{\mathcal{H}C}$. *Describing knowledge* of the form $((atmost\ R\ n)\ o)$, $((atleast\ R\ n)\ o)$, $((all\ R\ C)\ o)$ is mostly ignored in $\mathcal{W}_{\mathcal{H}C}$. Therefore, the closed-world-assumption (CWA) is valid w.r.t. $\mathcal{W}_{\mathcal{H}C}$, whilst the open-world-assumption (OWA) is valid w.r.t. $\mathcal{W}_{\mathcal{A}}$. In $\mathcal{W}_{\mathcal{A}+HC}$ the CWA and the OWA come into conflict, which implies that inconsistent universal axioms may be valid at the same time.

An inconsistency concerning to universal axioms $(C_1\ o)$ and $(C_2\ o)$ as described in definition 3.6 occurs when the TBox is inconsistent with the user-defined rules. The following example illustrates this effect:

Example 3.4:
Let A, B be concepts with $(A^{\mathcal{I}} \cap B^{\mathcal{I}}) = \emptyset$.

$\mathcal{W}_{\mathcal{A}} \models_{\mathcal{T}} (B\ o)$ is given.

If the user defines the rule

A(x) ← B(x).

the universal description $\mathcal{W}_{\mathcal{A}+\mathcal{HC}}$ satisfies the universal axioms $(A\ o)$ and $(B\ o)$ at the same time. This is contradictory to the TBox, because A and B are disjunct.

An approach to discover such potential inconsistencies is to extend the Horn clause program $\mathcal{HC}$ by *integrity rules.* The body of an integrity rule consists of predicates corresponding to disjunctive concepts, while the head consists of a special symbol.

Definition 3.8 (Extended Horn Clause Program) *Let $\perp$ be an 0-ary predicate. An* integrity rule *is a rule:*

$$\perp \leftarrow B_1(x), \ldots, B_m(x) \quad \textit{with} \quad \begin{array}{l} (B_1^{\mathcal{I}} \cap \ldots \cap B_m^{\mathcal{I}}) = \emptyset \\ B_1, \ldots, B_m \in \mathcal{C} \end{array}$$

$\mathcal{HC}'$ is a Horn clause program $\mathcal{HC}$ extended by integrity rules:

$$\mathcal{HC}' := \mathcal{HC} \cup \{\perp \leftarrow B_1(x), \ldots, B_m(x) \mid \begin{array}{l} B_1, \ldots, B_m \in \mathcal{C} \\ \wedge\ (B_1^{\mathcal{I}} \cap \ldots \cap B_m^{\mathcal{I}}) = \emptyset\ \} \end{array}$$

The *atomic-derivation-closure* of the extended Horn clause program contains the special symbol, if the Horn clause program is inconsistent.

Definition 3.9 (Inconsistent Horn Clause Program) *A Horn clause program $\mathcal{HC}$ is inconsistent, if the atomic-derivation-closure of the extended Horn clause program $\mathcal{HC}'$ contains the 0-ary predicate $\perp$:*

$\perp \in D^{\star}(\mathcal{HC}')$

4 RELATED WORK

The combination of rules and terminological reasoning offers two alternative system architectures.

In the first approach, the rule system dominates the terminological system. This paradigm is chosen in CLASP [9], where the basic component is a forward chaining rule system. Terminological knowledge is used during pattern matching and conflict resolution. The condition side of a rule matches a data item of the KB if the data item terminologically satisfies the condition side. The preference order of the applicable rules is determined by the terminological component.

The other alternative for combining rules and terminological reasoning is to integrate the rule component into a terminological reasoning system. In contrast to our approach, most known approaches use forward chaining rule systems. Some of them provide common production rules to achieve a higher expressive power. Unfortunately, the semantics of these approaches are rather unclear. For example, the rule component in LOOM [10] supports procedural attachment features in combination with user specified methods. Other approaches provide rules of a restricted form, e.g. CLASSIC [6] allows only to define rules of the form: $A(x) \rightarrow B(x)$, where A and B are concepts. Despite the additional expressive power is rather limited, the semantics of the rule inferences is unclear.

5 CONCLUSION

In this paper, we addressed the problem of integrating rule inferences into a terminological representation system. In contrast to other approaches, we use backward chaining rules to retrieve *additional knowledge* on *demand.* The rule component allows the use of n-ary relationships by introducing arbitrary n-ary predicates. In order to get a well-defined semantics, we restricted the form of our rules to Horn clauses. Our approach of integrating rules into a terminological representation system leads to inconsistencies. We have shown a solution which allows to detect some of the inconsistencies. We did not address the problem of how we can manage inconsistencies because this depends on the current application.

The proposed mechanism has been implemented in a terminological system, called TED&ALAN (Term Description Language and Assertional Language) [11].

Our approach has been influenced by our applications in the field of natural language processing and has been tested in a geographical query-answering system. It turns out that there is not only a need for terminological knowledge but rule-based knowledge is also important.

Our approach is based on a strongly restricted terminological formalism. However, the expressive power of KL-ONE-based systems used in practical applications has to be increased by additional description elements, e.g. *attributes*[12], *concrete domains*[13]. Therefore, we aim to examine our integration method with respect to such additional description elements.

REFERENCES

1. R. J. Brachman and J. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 2(9):45, 1985.

2. B. Owsnicki-Klewe. Configuration as a consisteny maintenance task. In W. Hoeppner, editor, *Proceedings of GWAI-88 - The 12th German Workshop on Artificial Intelligence*, pages 77–87. Springer Verlag, 1988.

3. J. Allgayer, K. Kobsa, C. Reddig, N. Reithinger, and D. Schmauks. Xtra: A natural-language access system to expert systems. *International Journal of Man-Machine Studies*, 31:161–165, 1989.

4. D.A. Cruse. *Lexical Semantics*, chapter 7, pages 157–180. Cambridge University Press, 1986.

5. P.F. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. MacGregor, W.S. Mark, D. McGuinness, B. Nebel, Schmiedel A., and J. Yen. Term subsumption languages in knowledge representation. *The AI Magazine*, 11(2):16–23, 1990.

6. Ronald J. Brachman. "reducing" classic to practice: Knowledge representation theory meets reality. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 247–258. Morgan Kaufmann, 1992.

7. J.-M. Nicolas. Logic for improving integrity checking in relational data bases. *Acta Informatica*, 18(3):227–253, 1982.

8. Stephen Muggleton and Cao Feng. Efficient induction of logic programs. In *FIRST CONFERENCE ON ALGORITHMIC LEARNING THEORY*, 1990.

9. John Yen, Robert Neches, and Robert MacGregor. Clasp: Integrating term subsumption systems and production systems. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):25–31, March 1991.

10. David Brill. *LOOM Reference Manual*, 1989.

11. Peter Forster and Gerrit Burkert. Term subsumption languages and lexical semantics. In Christof Peltason, Kai von Luck, and Carsten Kindermann, editors, *Proceedings of the Terminological Logic Users Workshop, KIT Project, Technische Universität Berlin*, pages 91–96, 1991.

12. Bernhard Hollunder and Werner Nutt. Subsumption algorithms for concept languages. Technical Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz, 1990.

13. Franz Baader and Philipp Hanschke. A scheme for integrating concrete domains into concept languages. Technical Report RR-91-10, Deutsches Forschungszentrum für Künstliche Intelligenz, 1991.

ADAPTIVE INFORMATION FILTERING BY MONITORING USER BEHAVIOR

Max Höfferer, Bernd Knaus, Werner Winiwarter

Institute of Applied Computer Science and Information Systems
Liebiggasse 4/3, A-1010 Vienna, Austria
e-mail: {mh, bk, ww}@ifs.univie.ac.at

ABSTRACT

Getting lost in e-mail spaces? This question arises from the increasing use of e-mail, which results in users being inundated by a huge stream of incoming documents. This paper presents a possible solution - CIFS. Our cognitive information filtering system consists of the following parts:

(1) linguistic analysis to obtain consistent representations of the contents of e-mails with the help of a morphological component and a component for generating postings,
(2) an evolutionary algorithm for prioritizing morphologically parsed messages, and
(3) a monitor to simulate cognitive user behavior.

1. INTRODUCTION

More and more users of *e-mail* and other on-line communication systems are faced with the problem of selecting relevant information in a space of different information-sources. Participants of on-line conferences, members of office-information systems, and network-diary makers get in trouble with an endless stream of messages. To overcome this *information overload problem* [1] information filtering techniques are being developed to deliver information to users.

State of the art IFS, e.g. *Information Lens System* [2], *EDS Template Filler* [3] or *Isceen* [4] offer a static behavior. A cognitive information system should have the ability to learn and adapt to the user's behavior.

The system presented - *Cognitive Information Filtering System* (*CIFS*) - applies genetic adaption to learn from user feedback and user behavior. CIFS distils e-mails from the input stream depending on the user's interests and evaluation judgements which are used to rank e-mail information.

With regard to efficiency the linguistic analysis of the e-mails is performed by use of a cascading architecture. The filtering proceeds by stepwise refinement of analysis techniques leading to a consecutive reduction of the problem of space. Therefore, only a very small fraction of the incoming e-mails must be analysed by sophisticated linguistic methods.

This paper is organized as follows. In Section 2 and Section 3 we will introduce the basic concepts of cognitive models and information filtering. The linguistic analysis applied consisting of the two modules indexer and parser will be presented in Section 4. Section 5 describes computational models of evolutionary processes. Finally, Section 6 gives a view of the architecture of the cognitive information filtering system.

2. COGNITIVE MODELS

In each electronic information source there can be so much detail that the information presented to the reader may be of lower quality and less relevant than traditional approaches. The ability to select information relevant to a user is essential to the viability of such services and requires an individual user model [5]. In our approach we have incorporated the following cognitive aspects to improve our system's ability for filtering e-mails.

Given the diversity of IFS users, the fact that they will not have the same problems or needs, and that the user's level of expertise and interest is likely to change in the cause of time, it is desirable that *profiles* be able to adapt to and support the requirements of individual users.

Monitoring data collection techniques, think-aloud protocols, tape recording of interaction, interviews, and questionnaires are helpful to understand the filtering process of an individual user [6]. The user's behavior is mapped on the behavior of the system.

The system must also have a model of itself in order to foresee its possible future actions and thus be able to choose the best way to act. Therefore we apply techniques similar to those used for debugging to give us a trace of system actions during the filtering process.

If no user interrupt occurs, in case of a relevant message, the system analyses any observations so that it can choose what to do with the next incoming similar message. Observation is needed to detect system actions during the filtering process. The observation me be passive or active depending on whether it memorizes what happened or makes experiments to find out autonomous new topics that may be interested for the user. In the first prototype we only use passive observation.

As long as the system has not noticed abnormal behavior, it reacts exactly alike without observing itself but simultaneously creates a trace of its actions and results; it may correct what goes wrong immediately or analyse it later. This observation may be continuous or occasional. In our approach we use *continuous observation*.

At first, the user may specify a catalogue of relevant topics (*user-profile*) or the system uses a *standard-profile* that determines what to observe. On getting more and more incoming messages the system learns by analysing the traces and the user's reaction to the output.

A complete cognitive user model has to represent the user's cognitive style and personality factors, the user's goals and plans, her or his capabilities and preferences, and the user's beliefs and knowledge. These characteristics are represented in a way that fits our genetic approach.

3. INFORMATION FILTERING

Information Filtering (IF) describes the processes of distribution and delivery of information to users of communication systems. Information filters assist users in finding relevant information but are also used to target information to potentially interested users. Information Filtering Systems (IFS) primarily handle unformated textual data like documents, semi-structured objects like electronic-mail messages (*e-mail*), NetNews articles, newswire stories or more complex structures like hypertext documents containing voice, graphics, and pictures. IFS process streams of incoming data based on descriptions of a single user or groups of users. These user "profiles" typically describe long-term interests [7] and individually depend on the way the user reacts to an incoming stream of information. The user can either select information items (positive kind of filtering) or remove items (negative kind of filtering).

IF is closely related to *Information Retrieval* (IR) which is concerned with the representation, storage, organisation, and accessing of information items such as documents [8]. The fundamental problem in IR is to identify the relevant documents from non relevant ones according to a particular user's request. There are three domains classifying IR research: indexing, retrieval and evaluation.

The *document representation* or *indexing* process performs the task of assigning information items to documents for purposes of retrieval. An indexing language maps the contents on documents to a textual representation.

The three main retrieval models in IR - *boolean, vectorspace* and *probabilistic model* - differ with respect to the matching process between user queries and document representations.

The *Boolean* model [8] compares queries and document descriptions by exact matching of index terms with the help of boolean operators. A disadvantage of the exact match model is that the whole document space is divided into two sets of relevant and non relevant documents with a ranking of documents according to a query.

In the *Vector Space* model [8] queries and documents are represented as vectors in a multi-dimensional space and compared with the help of statistical methods e.g. the *Cosine, Dice* or *Jaccard* function [9].

The *Probabilistic IR* model estimates the probabilities of a document's relevance by using the Bayes' theorem. The model is based on the *probabilistic ranking principle* (PRP) [10] which states that optimum retrieval is achieved when documents are ranked according to decreasing values of their probability of relevance with respect to the current query.

Differences between IF and IR are described in Table 1. The retrieval models described above are applied to IF [7], [11]. In "non-intelligent" IFS simple *Keyword Matching* determines whether the user's information interests match the incoming information items of the system.

	Information Filtering	*Information Retrieval*
System input	dynamic datastream	static database
User goals	long-term periodic desires	short-term intentions
User behavior to incoming data	reacts to	actively searching
Information processing	removing	finding (selection)
Information flow	distribution and organization	representation and organization
Use of the system	repeated	single
Representation of user interests	profiles	queries
Environment	more or less privacy	more or less public
User-groups	undefined	well-defined

Table 1. Information filtering vs. information retrieval

It is most important to the process of filtering that the indexing component consists of:

- a lexical scanner,
- a morphological component, and
- a component for generating postings.

4. LINGUISTIC ANALYSIS

Within CIFS linguistic analysis consists of two separate modules: the indexer and the parser. Additionally, a pre-filter reduces the amount of relevant e-mails. For this purpose, the pre-filter contains a set of keywords and phrases that initially describe the user's current interests. These descriptors weed out e-mails that are not about a *topic* of interest.

4.1 Indexer

As a second step within the filtering process the documents selected from the pre-filter are evaluated with regard to their relevance on the basis of an indexation module. The document texts are first transformed into a sequential word list. In order to retrieve the individual word boundaries well-approved heuristics are applied and abbreviations, compound words and special formats (e.g. time, date, or currency) are treated correctly by the use of a morphological pre-processor.

After generating the word list, all words that do not contribute to the meaning of the document (e.g. grammatical particles or expletives) are removed on the basis of a stop word list. An index is created which assigns each entry to a list of postings, that is, the positions of its occurrences. In order to check the equality of two words we do not apply an exact string match but we designed a special comparison module which applies simple techniques from morphological analysis to lemmatise the words concerned.

The developed tool is multilingual in the sense that techniques which are valid for any language are strictly separated from language-specific features. Furthermore, the latter are mapped to a high degree on parameters so that the adaption to a new language can be easily performed. Although we mostly applied approximate methods, we achieved a very high accuracy without losing too much speed. The following important morphological phenomena are analysed by our comparison module:

- *spelling errors:* Obviously, we do not consider all possibilities for the appearance of spelling errors but restrict ourselves to correct the most frequent error patterns such as insertion of one false character, the permutation of two letters or the omission of one character.
- *vowel-gradation:* This is an irregular morphological variation which occurs in many languages during the formation of inflexions (e.g. the ablaut in German) and can be resolved by using techniques from spelling error correction.
- *endings and suffixes:* The distinct endings of two words are compared with lists of legal inflexions and derivations which also gives a first evidence for the word category, an information which is essential for later syntactic analysis.
- *elision:* Elision is the omission of the unstressed e-sound, a frequent phenomenon in German and in Scandinavian languages.
- *binding sounds:* Finally, binding sounds tie together the individual parts of the word in the formation of derived or compound words.

For a more detailed discussion of morphological analysis in computational linguistics, especially for inflective languages, we refer to [12].

The resulting document index is matched against the user's profile by applying statistical similarity measures adopted from the vector space paradigm of information retrieval [8].

4.2 Parser

Only those documents which are evaluated to possess a sufficient similarity to the user's profile are subject of a more sophisticated syntactic and semantic analysis. Due to the requirements of information filtering with regard to processing time, natural language analysis can only be performed by *information extraction* and not by *text understanding.* This implies that only fractions of the document text are analysed, the retrieved information is mapped on some target representation and all other subtle aspects of meaning are left out of consideration [13].

Therefore, a cascading architecture is required which does not perform a complete linguistic processing for the whole document but narrows its scope by first retrieving text segments of special interest which can then be analysed more carefully. Within our filtering system this task of selecting interesting text segments is performed by the indexing module and is made available to the parser as a result of the match against the user's profile.

The contexts of these so-called *trigger words* are further analysed by the use of a simple yet efficient parsing algorithm in order to detect syntactic constructs (e.g. noun phrases, verb phrases or prepositional phrases). Only a straight-forward syntactic analysis is performed, all more tedious linguistic issues are ignored, according to the 'golden rule' of information extraction: 'to do the right amount of syntax, so that pragmatics can take over its share of the load' [13].

Pragmatics is modeled within our knowledge base by the use of frames as conceptual representation scheme. The interesting pieces of information contained in the analysed contexts are mapped to the slots of these frames in the cause of semantic analysis [14]. As a consequence of our main research objectives, the adaptive behavior of our information filtering tool, this knowledge base cannot be implemented by the use of a static structure. On the contrary, the represented knowledge must change dynamically in response to changes of the user's interests.

Finally, during discourse analysis all resulting frames are merged to obtain one consistent representation of the contents of an e-mail document. One difficult and important task of this merging process is to unify various interpretations, that is, to eliminate local ambiguities [15]. Once more it must be stressed that the final semantic representation will only model that part of contents that is relevant to the user, all other aspects are filtered out.

5. EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EA) use computational models of evolutionary processes as key elements in the design and implementation of problem solving systems [16]. A variety of evolutionary computational models have been proposed. They share a common conceptual base of simulating the evolution of individual structures via processes of *selection, mutation,* and *reproduction*. The processes depend on the perceived performance of the individual structures as defined by the environment. EAs maintain a population of structures which evolve according to rules of selection and other operators that are refered to as *search* (*genetic*) *operators* such as *recombination* and *mutation* (Fig. 1.). Each individual in the population receives a measure of its *fitness* in the environment. Reproduction - creation of a new individual from two parents - focuses attention on "high fitness" individuals, thus exploiting the fitness information available. Evolutionary computation (EC) includes research in the fields of *genetic algorithms* [17], *evolution strategies* [18], *artificial life* [19] and so forth.

The fitness of an individual measures how well an individual solves a task. This is important to the *genetic operators* to act. The crossover operator forms a new chromosome by combining parts of each of the two parent chromosomes. The selection operator evaluates the fitness of each chromosome and the mutation operator forms a new chromosome by making alterations to the values of genes in a copy of a single parent chromosome.

```
Procedure EA;
 (1) Start with an initial time
       $t := 0$;
 (2) Initialize a random population of individuals
       initpopulation $P(t)$;
 (3) Evaluate the fitness of all initial individuals in the population
       evaluate $P(t)$;
 (4) test for termination criterion (time, fitness, etc.)
   while not done do
    (a) increase time counter
          $t := t + 1$;
    (b) select a subpopulation for offspring production
          $P' :=$ selectparents $P(t)$;
    (c) recombine the genes of selected parents
          recombine $P'(t)$;
    (d) perturb the mated population
          mutate $P'(t)$;
    (e) evaluate its new fitness
          evaluate $P'(t)$;
    (f) select the survivors from actual fitness
          $P :=$ survive $P, P'(t)$;
   end while;
end EA;
```

Fig. 1. Evolutionary algorithm

EAs are sufficiently complex to provide robust and powerful *adaptive search mechanisms*. We use the EC approach to adapt descriptions of e-mails.

6. COGNITIVE INFORMATION FILTERING SYSTEM

Figure 2 shows the architecture of CIFS. The central component of the system, the cognitive information filter, contains a population of information objects (words, user usage patterns, phrases) called e-mail agents, a monitor, and a knowledge base.

The input to the filter module are the semantic representations of the e-mails. They form the initial population of the evolutionary algorithm, the so-called e-mail agents. E-mail agents obtain the following attributes: the user's evaluation, a bid in the range [+1, -1], a bid learning rate, and counters of frequency, recency and spacing.

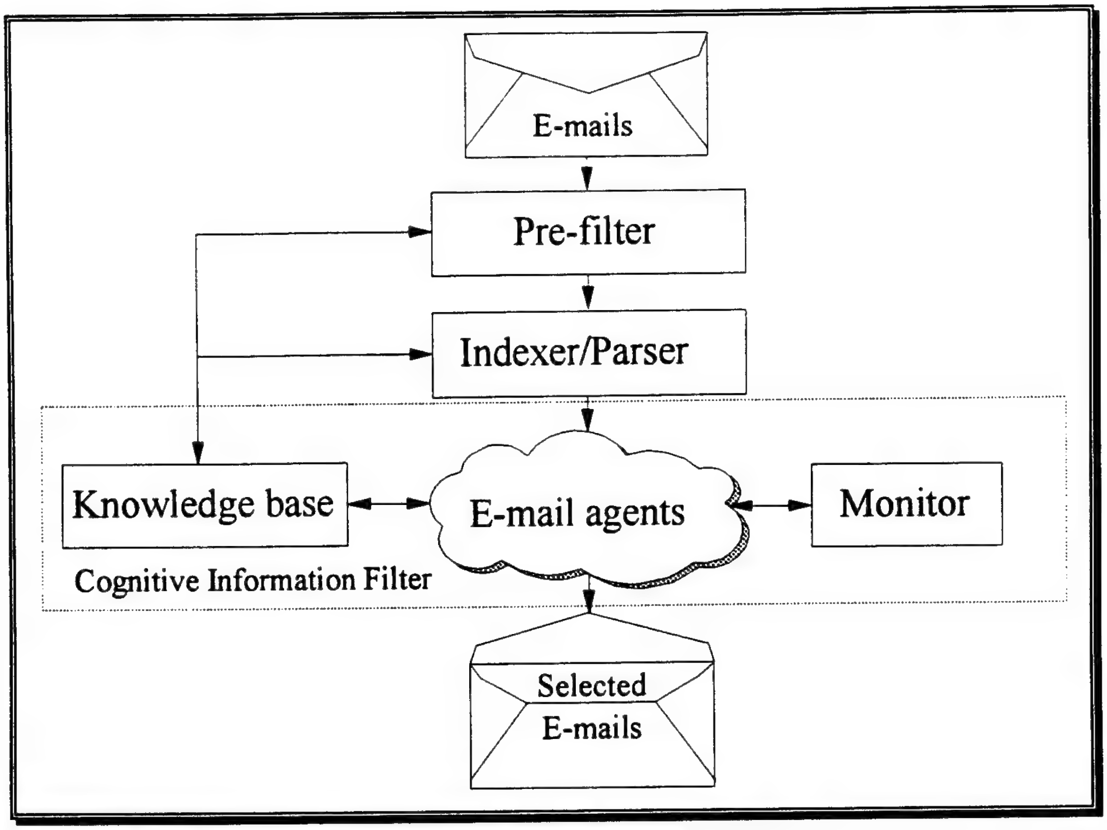


Fig. 2. Cognitive information filtering system

The learning algorithm in Figure 3 describes the genetic adaption where e-mail agents cooperate and compete for correct evaluations of the user's actual interest rating. E-mail agents learn by adjusting their evaluation, thus moving it closer to the user's evaluation. The payoff schema prevents the population from increasing.

Cognitive aspects are implemented as follows. E-mail agents that run out of strength leave the population for a calculated period of time - at that moment they are not relevant - and get incorporated into the pre-filter. Agents above the average fitness serve

- as 'new' keywords to the prefilter, e.g. the system reacts to the fact that a user has not been interested in a topic for a period of time, or
- remain in the population and get one more chance to be useful.

```
Repeat
(1) For any particular set of e-mail agents, measure the fitness (strength $S_i$) of each of its descriptions.
  (a)After an e-mail has been evaluated by a user, agents adjust their fitness:
      Calculate for each competing agent the new strength $S_i'$ after payoff:
      $S_i' = (S_i + LU)/(1 + L)$ with $L$ .. learning rate and $U$ .. user rating.
  (b) Enforce competition among $n$ e-mail agents by the payoff function [9]
      $P_i = E/n + 1/n - e_i - F$   $F$ .. fee
      error: $e_i = |R - S_i|$   $E = \sum e_i$
(2) After each user evaluation randomly select e-mail agents for reproduction.
      Cross the fittest agents to produce offsprings.
Until a predefined number of generations is obtained
```

Fig. 3. Genetic adaption of e-mail descriptions

The knowledge-base contains the semantic representation of the user profiles. Individual interests are mapped to frames. Their dynamic adaption is induced by e-mail agents .

The monitor component simulates a user's behavior and controls the agent's lifecycle. The monitor is a kind of feedback mechanism, measuring how effectively the history of usage patterns predicts current usage patterns and the probability that an item is needed given the history for such information [20]. Our system applies three time- and context-sensitive user preference functions: *Frequency* refers to the number of times information is needed within a specified period of time. *Recency* counts for the amount of time elapsed since the last need for an information item, and *spacing* refers to the distribution across time of the exposures to these information items. Equations employing these preference functions serve as predictors for the current need for an information item.

7. CONCLUSION

CIFS supports two general aspects:

- individual user preferences in daily operations with his or her e-mail system, and
- the actual contents of messages that are deemed interesting or uninteresting.

A further advantage is the time-saving device to cope with the information overload problem.

First tests on CIF to rate incoming e-mail messages from distinct *Internet* NetNews newsgroups of *comp.ai* (e.g. *comp.ai.nlang-know-rep*) showed promising results.

REFERENCES

1. G. Fischer, C. Stevens, "Information Access in comples, poorly structured information spaces," Proceedings CHI conference, pp.63-70, April 1991.
2. T. Malone, K. Grant, F. Turbak, S. Brobst, M. Cohen, " Intelligent Information-Sharing Systems," *CACM* 30(5), 1987, pp.390-402.
3. H.K. Shuldberg, M. Macpherson, P. Humphrey, J. Corley, "Distilling Information from Text: The EDS TemplateFiller System," *JASIS* 44(9), 1993, pp.493-507.
4. S. Pollock, "A Rule-Based Filtering System," *ACM TOIS* 6(3), 1988, pp.232-254.
5. R.B. Allen, "User models: theory, method, and practice," International Journal of Man-Machine Studies 32, 1990, pp.511-543.
6. J.Anderson, *Cognitive Psychology and its Implications,* A Series of Books in Psychology, Ed.: R. Atkinson, G. Lindzey, R. Thompson; New York: W.H. Freeman and company (1985).
7. N.J. Belkin, W.B. Croft, "Information Filtering and Information Retrieval: Two Sides of the Same Coin?," *CACM* 35(12), 1992, pp.29-38.
8. G. Salton, M.J. McGill, *Introduction to Modern Information Retrieval*, New York: McGraw Hill (1983).
9. T. Norault, M. McGill, M.B. Koll, "A performance evaluation of similarity measures, document term weighting schemes representations in a boolean system," Information retrieval Research, (eds). R.N. Oddy et.al., 1981, pp. 57-71.
10. S.E. Robertson, "The probability ranking principle in IR," *Journal of Documentation*, Vol. 33, 1977, pp. 294-304.

11. P.S. Jacobs, L.F. Rau, "SCISOR: Extracting Information from On-line News," *CACM* 33(11), 1990, pp.88-97.

12. W. Winiwarter, A.M. Tjoa, "Morphological Analysis in Integrated Natural Language Interfaces to Deductive Databases," Proceedings of the Fourth International Workshop on Natural Language Understanding and Logic Programming, Nara, Japan, Sep. 1993.

13. J.R. Hobbs, D.E. Appelt, M. Tyson, J. Bear, D. Israel, "Description of the Fastus System for MUC-4," Proceedings of the 4th Message and Understanding Conference, 1992, pp.169-177.

14. D. Ayuso, S. Boisen, H. Fox, H. Gish, R. Ingria, R. Weischedel, "BBN: Description of the PLUM System Used for MUC-4, " Proceedings of the 4th Message and Understanding Conference, 1992, pp.268-275.

15. A. Meyers, D. de Milster, "McDonnell Douglas Electronic System Company: Description of the TexUS System used for MUC-4, " Proceedings of the 4th Message a nd Understanding Conference, 1992, pp.207-215.

16. L.J. Fogel, J.W. Atmar (eds.), Proceedings of the first Annual Conference on Evolutionary Programming, Evolutionary Programming Society, San Diego, (1992).

17. D.E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison Wesley Inc., Reading, Mass. (1989).

18. H.P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, Birkhäuser Verlag, Basel, Stuttgart (1977).

19. T.S. Ray, "Is it alive, or is it a GA," Proceedings of the 1991 International Conference on Genetic Algorithms, 1991, pp.527-543.

20. J.R. Anderson, The Adaptive Character of Thought, Lawrence Erlbaum Associates: Hillsdale, New Jersey (1990).

REASONING ABOUT NUMBERS IN TECTON

– Preliminary Version —

Deepak Kapur*and Xumin Nie†
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, New York 12222, U.S.A.

Abstract

We discuss algorithms and heuristics for reasoning about numbers - rationals, integers, and naturals, and the associated operations $+$, $=$, $\geq$, in *Tecton*, a specification and verification system currently under development. A main objective is to make Tecton more efficient for tasks of reasoning about specification and programs that involve numbers. We extend Fourier's algorithm for deciding satisfiability of linear constraints over the rationals to linear constraints over the integers. This algorithm serves as a building block for a complete decision procedure for universally quantified Presburger arithmetic, in which a special emphasis is placed on deducing equalities and using them as rewrite rules for simplification and elimination. We discuss how this decision procedure has been integrated with definitions and properties of interpreted function symbols specified as terminating rewrite rules in the Tecton system.

1 Introduction

In the summer of 1991, we implemented a set of algorithms and heuristics to automatically reason about numbers – rationals, naturals and integers, in a verification and specification system *Tecton* [11] which is being developed on top of our theorem prover *Rewrite Rule Laboratory* (*RRL*) [15]. This paper is a belated report on these algorithms and our approach for implementing these algorithms into a rewrite rule based theorem prover. In particular, we extend Fourier's method for linear inequalities over the rationals, as explained in [20], to consider linear inequalities over the integers, and show its completeness as a decision procedure for satisfiability problem of linear inequalities over the integers.[1] The extended algorithm is used as a building block for a decision procedure for universally quantified Presburger arithmetic with uninterpreted symbols. A particular emphasis is placed on deducing equalities so that they can be used as rewrite rules for simplification and for detecting unsatisfiability.

We discuss how the decision procedure for Presburger arithmetic has been integrated with terminating (conditional) rewrite rules defining interpreted function symbols and their inductive properties. In particular, we discuss the interaction between the decision procedure and conditional

*Partially supported by National Science Foundation Grant No. CCR-9303394 and United States Air Force Office of Scientific Research Grant No. AFOSR-91-0361.

†Current address: Department of Computer Science, The Wichita State University, Wichita, Kansas 67208.

[1] After writing an earlier draft of this paper, we recently learned (May 1994) that Williams had already reported such an extension in [28]. A careful study of Williams's paper and Cooper's paper [4] would reveal that Williams was rediscovering Cooper's results and Presburger's procedure as reported in [6]. This is not very surprising because Fourier's method has been rediscovered many times in the mathematics literature.

rewriting while normalizing a formula to prove its validity in the Tecton system. Many examples are given to illustrate different decision procedures and their combinations.

A main objective in designing the Tecton system is to support construction of large complex proofs, such as those that typically are necessary in reasoning about specification and descriptions of generic components useful in software and hardware design [11]. Often, one needs to reason about natural numbers and integers, particularly in the context of linear data structures such as arrays and sequences, where indices are needed, and totally-ordered enumerative sets such as natural numbers and integers are natural choices for indices. Reasoning about numbers is also useful in reasoning about measures including size, length, depth, etc, defined on data structures. Using Presburger arithmetic, we have recently developed a method for checking completeness of function definitions defined on numbers using $0, s, +, \geq$ as terminating rewrite rules [10]. Using this method, it is possible to prove completeness of the function definitions used in an inductive proof of unique prime factorization theorem done on *RRL* using the cover set method for automating induction [31]; this method can also be used to prove the completeness of cover sets. In another paper, we have developed methods using Presburger arithmetic for generating induction schemes and other heuristics for enhancing the inductive theorem proving capabilities of *RRL* and Tecton [14].

Algorithms for deciding properties of numbers were extensively discussed in the literature in the mid-70's and early 80's when there was considerable interest in program verification. A particular focus has been on a decision procedure for a subclass of *Presburger arithmetic* (in particular, universally quantified theory of Presburger arithmetic). Cooper [4] gave an algorithm which improved upon on his previous algorithms as well as an algorithm given in logic text-books, e.g. [6]. Shostak [25] proposed a method for proving formulas in Presburger arithmetic using a procedure for solving linear inequalities over the rationals using the ***sup-inf*** method proposed by Woody Bledsoe [1, 2]. Subsequently, Shostak [26] gave an algorithm for Presburger arithmetic with uninterpreted function symbols, formulas handled by Cooper's algorithm; this algorithm was based on integer linear programming. Nelson and Oppen [23] proposed an elegant framework of cooperating decision procedures with a simplex based algorithm for solving linear inequalities over the rationals as one of the decision procedures in the Stanford Pascal Verifier. This framework was adopted in the Eves system and has been implemented in its theorem prover [5].

In the mid-80's, Boyer and Moore [3] incorporated a rational-based procedure in their theorem prover for automating proofs by induction.[2] They extensively discussed their implementation, in particular representations used for building a data base for rewriting using contexts, as well as specific techniques used to manage the interaction between the rewriter and the decision procedure.

Lassez and his group have been investigating efficient methods for linear constraints over the rationals in the context of constraint logic programming [19, 20]. In particular, they have revived interest in Fourier's algorithm for checking the satisfiability of linear constraints over the reals and the rationals. In [20], Lassez and Maher discussed Fourier's algorithm in detail and showed how it could be used to deduce implicit equalities in inequality constraints. The discussion of Fourier's algorithm below is based on [20].

We have recently learned of another interesting application of reasoning methods over numbers in the area of data dependency analysis for compilers for supercomputers. Presburger arithmetic has been used to study dependency among reads and writes of array elements in loop programs. For more details, the reader can consult [24].

2 Algorithms for Reasoning about Linear Inequalities

The main algorithm for solving linear inequalities that we implemented is that of Fourier [20], which as Boyer and Moore remarked [3], "is just a formalization of the high school idea of "cross multiplying and adding" equalities to eliminate variables." In this section, we first review Fourier's algorithm

[2] We believe that Hodes' method [7] implemented in Boyer and Moore's prover is essentially Fourier's algorithm (perhaps with minor variations).

for checking satisfiability of linear inequalities over the rationals, and then describe how Fourier's algorithm can be extended to derive *implicit equalities* [20]. Later, we discuss our extension of Fourier's algorithm to linear integer inequalities. This extension serves as a basis for a decision procedure for universally quantified Presburger arithmetic.

Henceforth, by *Presburger arithmetic*, we mean the universally quantified theory built using natural number constants (or integer constants whenever that is evident from the context), variables over the natural numbers (or integers), addition (and subtraction), the usual arithmetical relations (such as $=, \geq, >, \leq, <$), and the usual first order logical connectives; no other symbols are assumed. By a *Presburger formula*, we mean a quantifier-free formula in Presburger arithmetic. Later we consider extended Presburger arithmetic in which function symbols are introduced; if a function symbol is not constrained using any additional properties, it is considered to be *uninterpreted*; otherwise it is considered to be *interpreted*. Following [20], we now introduce some terminology.

Definition 2.1 An (integral) *inequality* is an expression of the form $\sum_{j=1}^{n} a_j x_j \leq b$ where a_j and b are integers. Let C be an inequality $\sum_{j=1}^{n} a_j x_j \leq b$, we use $C^{=}$ to denote the equation $\sum_{j=1}^{n} a_j x_j = b$. For $\alpha > 0$, αC is the inequality obtained by multiplying C by α, i.e. $\sum_{j=1}^{n} \alpha a_j x_j \leq \alpha b_k$. The sum $C_1 + C_2$ of inequalities C_1 $(\sum_{j=1}^{n} a_{1j} x_j \leq b_1)$ and C_2 $(\sum_{j=1}^{n} a_{2j} x_j \leq b_2)$ is the inequality $\sum_{j=1}^{n}(a_{1j} + a_{2j})x_j \leq (b_1 + b_2)$. If an inequality H can be expressed as $H = \sum_{k=1}^{m} \alpha_k C_k$ and $\alpha_k \geq 0$ $(1 \leq k \leq m)$, then H is called as a *non-negative linear combination* of inequalities C_1, ..., C_m. When each α_k is strictly positive, then H is a positive linear combination.

We define in the usual way the concepts of satisfiability of formulas over a domain and of logical consequence ($\models$). We define an *implicit equality* in a domain in a set $\mathcal{P}$ of inequalities to be an equation $C^{=}$ where C is an inequality in $\mathcal{P}$ and $\mathcal{P} \models C^{=}$ over the domain. A domain could be rationals, integers or naturals. Often, the domain would be evident from the context.

2.1 Fourier's algorithm for rationals

Let $\mathcal{P}$ be a set of inequalities, and let x be a variable appearing in it. Let C_i be an inequality in $\mathcal{P}$ with a negative coefficient $-c_i$ of x, $c_i > 0$, and C_j be an inequality in P with a positive coefficient c_j of x, $c_j > 0$. Then $c_i C_j + c_j C_i$ is an inequality which does not contain x. Such an operation is called *elimination,* since it eliminates an occurrence of the variable x from C_i, C_j. Every common solution of C_i, C_j is also a solution of $c_i C_j + c_j C_i$; further, every solution of $c_i C_j + c_j C_i$ can be extended to get a common solution of C_i, C_j. Because of such relationship among the solution of the derived inequality to C_i, C_j, i.e. a common solution of C_i, C_j can be projected by forgetting the value of x to get a solution of $c_i C_j + c_j C_i$, this operation is also called *projection.*

Given $\mathcal{P}$, we can derive from $\mathcal{P}$, all inequalities that do not have any occurrence of x by pairing inequalities containing x with negative coefficients with inequalities containing x with positive coefficients. The set of all such inequalities, together with the inequalities of $\mathcal{P}$ which do not contain x, is the result of a *macro Fourier step* eliminating x completely from $\mathcal{P}$. Typically many redundant inequalities are generated which can be detected and deleted using the test that $C \leq c$ implies $\alpha C \leq d$ for any $\alpha > 0, \alpha c \leq d$.

If x does not have a positive coefficient in any inequality of $\mathcal{P}$ or x does not have a negative coefficient in any inequality of $\mathcal{P}$, then the Fourier step eliminating x from $\mathcal{P}$ simply deletes all inequalities containing x. This is because an appropriate value of x can be constructed to satisfy all such inequalities. Such a Fourier step is called *trivial.*

Fourier's algorithm consists of repeatedly performing macro Fourier steps, eliminating one variable at a time, until either the set contains a contradictory inequality $0 \leq c$ where c is negative, implying that the original set of inequalities is unsatisfiable, or all variables are eliminated without yielding any contradictory inequality, thus implying that the original set of inequalities is satisfiable.

It is easy to see that any inequality C produced in Fourier's algorithm is a non-negative linear combination of inequalities $C_1, \ldots, C_m \in \mathcal{P}$. The following theorems from Käufl ([16]) and Lassez and Maher ([20]) serve as a basis for deriving implicit equalities from a set $\mathcal{P}$ of inequalities.

Theorem 2.2 *If $\mathcal{P}$ implies an equality e, there is a finite subset $\{C_1, ..., C_k\}$ of $\mathcal{P}$ such that $\mathcal{P} \models e$ iff $\{C_1^=, ...C_k^=\} \models e$* ([16]).

Theorem 2.3 *If $\sum_{k=1}^{m} \alpha_k C_k = 0 \leq 0, C_k \in \mathcal{P}, \alpha_k > 0, 1 \leq k \leq m$, then each $C_j, 1 \leq j \leq m$, is an implicit equality* ([20]).

Theorem 2.4 *An inequality C_k in a set of consistent inequalities $\mathcal{P}$ is an implicit equality in rationals iff Fourier's algorithm produces an inequality $0 \leq 0$ using C_k* ([20]).

2.2 Extending Fourier's algorithm to integers

Fourier's algorithm can be extended to form a complete decision procedure for checking satisfiability of linear inequalities over the integers. It is easy to see that if a set of inequalities is unsatisfiable over the rationals, it is unsatisfiable over integers. It is however possible for a set of inequalities to be unsatisfiable over the integers, but it may have a satisfying assignment over the rationals. A simple example is that of $\{2x \leq 1, -2x \leq -1\}$.

In the extended Fourier's algorithm the basic Fourier step is the same, i.e. given an inequality C_i in $\mathcal{P}$ with a negative integral coefficient $-c_i$ of x, $c_i > 0$, and C_j, another inequality in $\mathcal{P}$, with a positive integral coefficient c_j of x, $c_j > 0$, $d_i C_j + d_j C_i$ is an inequality wich does not contain x, where $d_i = \frac{lcm(c_i,c_j)}{c_j}, d_j = \frac{lcm(c_i,c_j)}{c_i}$. It would have been okay to multiply C_j by c_i and C_i by c_j, but one is likely to see smaller integers if we remove the gcd of c_i, c_j from the multipliers.

If during the algorithm, while eliminating a variable x, an inequality of the form $0 \leq c_i$, where c_i is positive is generated, then in contrast to the rational case, there is no guarantee any more that inequalities leading to this inequality can be satisfied. Unlike in the rational case, it may not be possible to extend an integer solution of the derived inequalities (which does not have any assignment for x) to an integer solution of the original inequalities, as the following simple example illustrates. Let $C_i = 4x \leq 7, C_j = -6x \leq -8$. We deduce the inequality: $3 * 4x - 2 * 6x \leq 3 * 7 - 2 * 8$, which is equivalent to $0 \leq 5$. The two inequalities being considered are: $12x \leq 21$ and $-12x \leq -16$, i.e. $16 \leq 12x \leq 21$. But there is no x that lies in this interval. So if after eliminating all the variables, no contradictory inequality is generated, there is no guarantee yet that the inequalities are satisfiable. It becomes necessary to check whether intervals for variables are large enough to produce a satisfying assignment.[3]

We process equalities first. For each equality, we check whether the gcd of the coefficients of the variables divides the constant in the equality; if it does not, then the equality cannot be satisfied. In an equality, a variable with a unit coefficient is preferred for elimination. A system of equalities is solved using the algorithm given in [17]. Variables solved for can be eliminated from the inequalities.

A constraint $c_1 x_1 + \cdots c_l x_l \leq d$, where d is an integer, can be simplified by dividing it by $g = gcd(c_1, \cdots, c_l)$ to give $c_1' x_1 + \cdots c_l' x_l \leq d'$, where $c_i' = \frac{c_i}{g}, d' = \lfloor \frac{d}{f} \rfloor$.

Below, we assume that equalities have been processed and inequalities have been simplified. Any new equality or inequality generated during the extended Fourier procedure is preprocessed in the same way.

As in [20], given a set $\mathcal{P}$ of integer inequalities, classify all inequalities based on whether x appears with a negative coefficient, positive coefficient, or does not appear at all.

$$\begin{array}{ll} l_i \leq c_i x & i = 1, 2, \ldots, p, \\ d_j x \leq r_j & j = 1, 2, \ldots, q, \\ g_l \leq 0 & l = 1, 2, \ldots, s. \end{array}$$

[3] For unsatisfiability, an additional constraint can be added: if C_i is equal to $A \leq c_i x$, C_j is equal to $c_j x \leq B$, add $d_i B < d_j A + lcm(c_i, c_j) - d_i - d_j$.

Let $\mathcal{P}'$ be the set of inequalities deduced from $\mathcal{P}$ by eliminating x by pairwise resolution of an inequality with a negative coefficient of x and an inequality with a positive coefficient of x.[4] Assuming that $\mathcal{P}'$ is satisfiable and σ is a satisfying assignment for $\mathcal{P}'$, it is possible to extend this assignment to include a value of x to have a satisfying assignment for $\mathcal{P}$ provided the value lies in the intervals specified by $\mathcal{P}$. This check is delayed until the end; however, it is necessary to keep track of intervals for possible values that x can take.

Let $L = lcm(c_1, \ldots, c_p, d_1, \ldots, d_q)$. Then inequalities in $\mathcal{P}'$ imply these inequalities[5]

$$\begin{array}{ll} \frac{L}{c_i} l_i \leq \frac{L}{d_j} r_j & i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q, \\ g_l \leq 0 & l = 1, 2, \ldots, s. \end{array}$$

If $\mathcal{P}'$ has a solution σ, then σ can be extended to include a value of x that would be a solution of $\mathcal{P}$ provided an integral multiple of L lies in the interval

$$\left[max\{\sigma(\frac{L}{c_i} l_i)\}, min\{\sigma(\frac{L}{d_j} r_j)\} \right], i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q.$$

Let a finite set of inequalities, $\{\frac{L}{c_i} l_i \leq L\ x \leq \frac{L}{d_j} r_j,\ \ i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q\}$ be called the *defining constraint* on x, denoted as Def_x. If for every solution σ of $\mathcal{P}'$, no integral multiple of L lies in the interval defined by instantiating the defining constraint of x by σ, then $\mathcal{P}$ is unsatisfiable even if $\mathcal{P}'$ is satisfiable. For checking satisfiability, it then becomes necessary to generate an assignment at the end of the algorithm and extend it by making sure that every variable satisfies its defining constraint. While generating a satisfying assignment, we may find that it is not possible to extend a partial assignment further, thus detecting unsatisfiability.

If a Fourier step involves deleting inequality constraints because a variable x being eliminated appears only with negative coefficients (or positive coefficients), say, $l_i \leq c_i x,\ c_i > 0,\ 1 \leq i \leq p$ ($d_j x \leq r_j,\ d_j > 0,\ 1 \leq j \leq q$, respectively), then $max\{\frac{L}{c_i} l_i \mid 1 \leq i \leq p\} \leq Lx$ ($Lx \leq min\{\frac{L}{d_j} r_j \mid 1 \leq i \leq p\}$, respectively) serves as the defining constraint for x.

Theorem 2.5 *Let $\mathcal{P}$ be a finite set of inequalities as defined above. Let $\mathcal{P}'$ be the inequalities generated from $\mathcal{P}$ after eliminating x. Let Def_x, the defining constraint for x, be*

$$\frac{L}{c_i} l_i \leq L\ x \leq \frac{L}{d_j} r_j,\quad i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q.$$

$\mathcal{P}$ is satisfiable if and only if $\mathcal{P}'$ is satisfiable using an assignment σ and there exists a multiple of L in the interval $\left[\ max\{\sigma(\frac{L}{c_i} l_i)\},\ min\{\sigma(\frac{L}{d_j} r_j)\}\ \right]$.

Example 2.6 Let $\mathcal{P} = \{4x + 4y \leq 3, 2x + 2y \geq 1\}$. After a Fourier step to eliminate x, one gets $2 - 4y \leq 3 - 4y$, where $L = lcm(2, 4) = 4$, which gives the inequality $2 \leq 3$, which is satisfiable. The defining constraint for x is $2 - 4y \leq 4x \leq 3 - 4y$, and there is no integral multiple of 4 in the interval $[2 - 4y, 3 - 4y]$. Hence $\mathcal{P}$ is unsatisfiable.

In the above example, it could be deduced quickly that no value of x can satisfy the inequalities in $\mathcal{P}$. In general, many macro Fourier steps may have to be done, and then solutions tried to get to an interval using which a solution may not be extensible.

There are many refinements and optimizations for checking satisfiability. For example, it is not necessary to search through all possible assignments of a variable in the whole interval of values

[4] For simplicity, we assume in the discussion below that exactly one variable x is eliminated in a macro Fourier step. The case of more than one variables getting eliminated while eliminating one variable can be handled in a similar manner.

[5] Because of pairwise resolution, inequalities generated in $\mathcal{P}'$ are likely to have smaller numbers appearing as coefficients. Cooper [4] instead multiplied every inequality with L/c_i, where c_i is the absolute value of the nonzero coefficient of x in an inequality C_i, replaced Lx by a new variable x' and required x' to be divisible by L.

given by its defining constraint. We will not discuss them here for lack of space. Neither will we discuss the more general case of a linear combination of variables possibly getting eliminated in an elimination step and its implications for checking satisfiability. An interested reader may consult [13]. We would however like to mention that delaying the interval check to the end is useful for unsatisfiable formulas. Although we have not performed an experimental comparison, it is likely to be much more efficient than Cooper's method in which the defining constraint on a variable is expressed using *mod*.

Recently, Jaffar et al [9] discussed a transitive closure algorithm for processing integer inequalities that have at most two variables; this algorithm is based on Shostak's method for computing loop residues. They showed that if the coefficients of variables in these inequalities are units (TVPI), then satisfiability can be checked in $O(n^3)$ time and $O(n^2)$ space, where n is the number of inequalities. Fourier's algorithm can be used for unit *TVPI* problems, and it has the same complexity.

2.3 Using implicit equalities in detecting unsatisfiability

In contrast to Cooper's approach as well as Shostak's approach, a rewrite rule based prover is always on the lookout for deducing equalities so that they can be used for simplification, rewriting, and eliminating variables. *RRL* uses equalities (implicit as well as explicit) as rewrite rules. As soon as such an equality is identified, it can be used as a rewrite rule to simplify other inequalities. This becomes especially important while using the extended Fourier's algorithm for deciding linear arithmetic with uninterpreted function symbols as well as in the presence of function symbols defined using rewrite rules. Below we discuss how implicit equalities are derived using the extended Fourier's algorithm for integer inequalities.

The method of Lassez and Maher discussed earlier for the case of linear inequalities over the rationals extends to integers as the following theorem states.

Theorem 2.7 *An inequality C_k in a set of consistent inequalities $\mathcal{P}$ is an implicit equality in integers if the extended Fourier's algorithm produces an inequality $0 \leq 0$ using C_k.*

Example 2.8 Consider the set of integer linear inequalities

$$\begin{array}{llllll} C_1: & a-b \leq 0, & C_2: & b-f(a) \leq 0, & C_3: & f(a) \leq 1, \\ C_4: & -a-b \leq -2, & C_5: & -b-f(b) \leq -2, & C_6: & f(a)-f(f(b)) \leq -1. \end{array}$$

It is easy to verify that

$$C_1 + 2C_2 + 2C_3 + C_4 = 0 \leq 0$$

Thus we have

$$\begin{array}{llll} C_1^=: & a = b, & C_2^=: & b = f(a), \\ C_3^=: & f(a) = 1, & C_4^=: & a + b = 2. \end{array}$$

As will be clear from subsequent discussion, the above equalities are used to deduce that each of $a, b, f(a), f(b), f(f(b))$ take the value 1. From this, the unsatisfiability of the above set of inequalities follows since C_6 cannot be satisfied.

The reader should note two crucial properties which are different from the case of inequalities over the rationals. Firstly, there is no guarantee that the deduced implicit equality is satisfiable, as the inequalities from which the equality is deduced may not be satisfiable. The second is that the converse of the above theorem does not hold. That is, there are equalities (implicit as well as derived) of $\mathcal{P}$ which cannot be deduced directly by the extended Fourier's algorithm. It is necessary to do some extra work, as illustrated by the following examples.

Example 2.9 Consider the following set of integer linear inequalities

$$\begin{array}{llll} C_1: & x-4y \leq -1, & C_2: & -x+4y \leq 2, \\ C_3: & x \leq 2, & C_4: & -x \leq -1. \end{array}$$

Eliminating y from C_1, C_2, we obtain $0 \leq 1$; the defining constraint for y is that $x + 1 \leq 4y \leq x + 2$. Eliminating x from C_3, C_4, we obtain $0 \leq 1$; the defining constraint for x is that $1 \leq x \leq 2$. So the extended Fourier's algorithm does not deduce any equality. However, the above set of inequalities is satisfiable, and the only solution is $x = 2, y = 1$ since for $x = 1$, there is no y satisfying the defining constraint of y. This implies that $4y = x + 2$ as well as $x = 2$ are implicit equalities. This example illustrates that even though the constraint on x suggests two assignments, one of those assignments cannot be extended as there are no values of y satisfying the defining constraint of y.

Example 2.10 Consider the following set of integer linear inequalities

$$\begin{array}{llll} C_1: & x - 4y \leq -1, & C_2: & -x + 4y \leq 2, \\ C_3: & x \leq 3, & C_4: & -x \leq -2. \end{array}$$

The defining constraint for y is that $x+1 \leq 4y \leq x+2$, and the defining constraint for x is $2 \leq x \leq 3$. For both values of x, $y = 1$ which is an equality that follows but cannot be obtained directly from any inequality by replacing $\leq$ by $=$. This example illustrates that a finite set of linear inequalities may not have any inequality which is really an equality but it may imply other equalities.

Extending Fourier's algorithm to deduce all implicit equalities from linear inequalities over the integers is an interesting challenge.

2.4 A decision procedure for pure linear arithmetic

A quantifier-free formula without any (uninterpreted or interpreted) function symbols can be decided using the extended Fourier's algorithm. Given a formula F, its proof can be done by refutation. An easy (but inefficient) way is to transform the negation of F into a disjunctive normal form,

$$\neg F \equiv G_1 \vee G_2 \vee \cdots \vee G_n$$

where each G_i is a conjunction of integer inequalities.[6] Henceforth, a disjunctive normal form will be used for the sake of simplicity. The formula $\neg F$ is satisfiable iff one of G_i is satisfiable, and so F is valid iff none of G_i is satisfiable.

Every literal is transformed into a disjunction of conjunction of inequalities using only $\leq$ predicate. For example, $A \neq B$ is replaced by $(A > B \vee A < B)$; in case of naturals and integers, $A < B$ is replaced by $A + 1 \leq B$, whereas for rationals, a new variable, called a *surplus variable*, is introduced, so we have $A + z \leq B$ with the requirement that z is strictly positive. Firstly, equalities are solved and used to eliminate variables. Inequalities are simplified. Then Fourier's extended procedure is applied on inequalities. Any equalities and inequalities deduced are first preprocessed with redundant inequalities removed.

It should be noted that one need not have three separate implementations of Fourier's algorithms - one for rationals, another for integers and third for naturals. Instead, it suffices to have an implementation of the extended Fourier's algorithm. In the case of integers or naturals, satisfiability is declared only if no contradictory inequality is generated and a satisfying assignment for variables can be generated from their defining constraints. In the case of rationals, satisfiability is declared if no contradictory inequality is generated. Further surplus variables are eliminated at the end, and before elimination, it is checked that each surplus variable can indeed be assigned a positive value (which is ensured by checking that the interval for the surplus variable includes a positive value); one way to check this is to analyze the minimum of upper bounds for positiveness.

For natural numbers, an additional constraint of nonnegativeness for every variable over the naturals is added. But more importantly, the semantics of subtraction, to be denoted by $\ominus$ (also known as Peano subtraction), is different. Given $x \ominus y$, there are two cases to be considered: (i) x is no smaller than y, and (ii) x is smaller than y, in which case the result is 0.

[6] As pointed by Cooper [4], it is not necessary to transform a formula into a disjunctive normal form; instead once negation is pushed down to the literals, the resulting formula can be checked for satisfiability as it is. Another approach toward avoiding generating a disjunctive normal form is to use *if-then-else* notation used in the AFFIRM system as well as Boyer and Moore's theorem prover.

3 Linear Arithmetic with Uninterpreted Function Symbols

Given a Presburger formula G with occurrences of uninterpreted function symbols, a formula G_a is generated, which is a conjunction of $GE \wedge G'$, such that G' is a pure Presburger formula without any uninterpreted symbols and GE is a conjunction of equalities relating functional terms to new symbols, called *abstraction constants*. GE does not include any occurrence of any arithmetic operator other than $=$. G_a is unsatisfiable if and only if G is unsatisfiable.

Equalities in G' are used to eliminate variables from inequalities in G' as well as from GE. The resulting equalities in GE can be processed using a congruence closure algorithm [23] or a Knuth-Bendix completion procedure on ground terms which is guaranteed to terminate [18, 8, 12]. In *RRL* , the Knuth-Bendix completion procedure on ground terms is used which gives a complete decision procedure for ground equalities as a finite set of rewrite rules. The rewrite rules are used to normalize G'.

The formula G' is subsequently processed using the extended Fourier's algorithm. Any equality deduced from G' is added to GE and a new canonical ground rewrite system is incrementally generated by adding the new equalities. This may result in additional equalities relating functional terms and hence abstraction constraints, which may further simplify the inequalities currently being considered in the extended Fourier algorithm (see example 2.11 for instance in which implicit equalities are used to relate terms).

This repeated interaction between the decision procedure for equality on ground terms, equalities deduced from Fourier's algorithm and normalization of inequalities using equalities terminates. This is so because (i) the ground completion procedure always terminates and (ii) Fourier's algorithm terminates since in each macro step, at least one variable appearing in the inequalities is eliminated (and number of variables appearing in the inequalities never increases; the number may decrease because of equalities).

The soundness of the procedure follows from the soundness of the ground completion and Fourier's algorithm. So, if unsatisfiability is detected by Fourier's algorithm, then G is unsatisfiability. Otherwise, if the procedure terminates giving a satisfying assignment for G_a, then G can be shown to be satisfiable by building a model for it provided all equalities that can be deduced from GE and G' have been made explicit using Fourier's algorithm and ground completion. Assuming that there is a way to extend Fourier's algorithm to deduce all implicit equalities, this approach gives a complete decision procedure for Presburger arithmetic with uninterpreted symbols. In the absence of a procedure for deducing all implicit equalities from integer inequalities, a method proposed in [23] can be used. All possible equalities among variables appearing in G_a are deduced using a complete decision procedure for unsatisfiability of integer inequalities.

Example 3.1 Consider an example from [27]:

$$z = f(x-y) \wedge x = z + y \wedge -y \neq -(x - f(f(z))).$$

The functional terms $f(x-y)$, $x-y$, and $f(f(z))$ are abstracted to be u_1, u_2 and u_3, respectively. We thus have:

$$(f(x-y) = u_1 \wedge x - y = u_2 \wedge f(f(z)) = u_3) \wedge (z = u_1 \wedge x = z + y \wedge -y \neq -(x - u_3)).$$

Ground completion on equalities would give: $\{z \rightarrow u_1, x \rightarrow y + u_1, u_2 \rightarrow u_1, u_3 \rightarrow u_1, f(u_1) \rightarrow u_1\}$. ¿From this and $u_1 \neq u_3$, a contradiction follows.

4 Integration of Fourier's Algorithm in a Rewrite System

Most work on Presburger decision procedure assumed formulas in pure Presburger arithmetic [1, 6] or formulas with uninterpreted function symbols [4, 2, 25, 26]. In [27, 23], methods for combining decision procedures are described which enable handling some interpreted function symbols, for example the quantifier-free theory of *lists* with *cons, car, cdr.* In [3], Boyer and Moore described how Hodes' procedure can be integrated with interpreted symbols defined as lisp functions. In this section, we discuss the integration of a complete decision procedure for Presburger arithmetic with interpreted function symbols defined as a finite set of canonical unconditional rewrite systems.

Definition 4.1 A *rewrite rule* is an oriented equation of the form $lhs \rightarrow rhs$, where lhs is its left-hand side and rhs is its right-hand side. A *rewrite system* $\mathcal{R}$ is a finite set of rewrite rules.

Given a rewrite system $\mathcal{R}$ and a term $t[t']$, where t' is a subterm of t, $\mathcal{R}$ reduces t to another term s if there is a rewrite rule $l \rightarrow r$ and a substitution γ such that $\gamma(l) = t'$; s is t with the subterm t' replaced by $\gamma(r)$.

Definition 4.2 A rewrite system $\mathcal{R}$ is *canonical* if and only if it is *terminating* and *confluent*, i.e. it does not admit any infinite rewrite sequence and every rewrite sequence from a term t can be extended to give a unique normal form, called the *canonical form* of t.

It is easy to see that $\mathcal{R}$ is a decision procedure for equations. Given an equation $s = t$, we compute the canonical forms of s and t, and check for equality. If they are equal, then $\mathcal{R} \models s = t$; otherwise, $\mathcal{R} \models (s = t)$ does not follow. However, it is not possible to get, in general, a decision procedure for the quantifier-free theory of $\mathcal{R}$ even though $\mathcal{R}$ is canonical. For example, consider $\mathcal{R}$ to be the associativity rule (oriented in any way), which constitutes a canonical rewrite system and serves as a decision procedure for the equational theory of free semi-groups. If it was possible to get a decision procedure for the quantifier-free theory of $\mathcal{R}$, then we would be able to solve the word problem of any finitely presented semi-group (including the ones with unsolvable word problems) by specifying it as a conditional equation, in which the conditions are the finite presentation of the semi-group, and the conclusion is an equation relating two words.

The Knuth-Bendix completion procedure and its extensions can be used as a semi-decision procedure for the quantifier-free theory of $\mathcal{R}$. Whether

$$\mathcal{R} \models F = ((s_1 = t_1 \wedge \cdots \wedge s_k = t_k) \supset (s = t))$$

can be checked as follows: let s', t', s'_i, t'_i be Skolem forms of s, t, s_i, t_i, respectively, obtained by introducing Skolem constants for the variables in F. The completion procedure can be attempted on $\mathcal{R} \cup \{s'_i = t'_i\}$. If the conclusion $s' = t'$ in the conditional equation indeed follows, while attempting to generate an augmented canonical system from $\mathcal{R}$, the conclusion can be proved.

If $\mathcal{R}$ is such that for any finite set GE of ground equations expressed using symbols in $\mathcal{R}$ and constants, completion terminates producing a finite canonical rewrite system $\mathcal{R}'$, then $\mathcal{R}'$ can be used to decide whether the conclusion of a conditional equation whose conditions constitute GE, follows from $\mathcal{R}$ or not. We will call a canonical $\mathcal{R}$ with this property as an *admissible* rewrite system $\mathcal{R}$. For an admissible $\mathcal{R}$, its quantifier-free theory is decidable using completion. (The above statement is true even when the conjecture above is generalized to $(L_1 \wedge \cdots \wedge L_k) \supset L$, where L as well as each $L_i, 1 \leq i \leq k$, is an equation or a disequation.)

Example 4.3 Consider an interpreted symbol defined using the equation $f(f(x)) = x$. The rewrite system $\{f(f(x)) \rightarrow x\}$ can be shown to be admissible. The conjecture is $f(x) = f(y) \supset x = y$. The Skolem form of its negation is $f(a) = f(b) \wedge a \neq b$. The ground equality $f(a) = f(b)$ is oriented to $f(a) \rightarrow f(b)$. Its left side $f(a)$ superposes with $f(f(x))$ giving a ground superposition $f(f(a))$ from which $a = b$ follows, and this gives a contradiction, implying that the conjecture follows.

This example indicates that it is not sufficient to normalize literals in a conjecture using $\mathcal{R}$; instead it is necessary to superpose ground equalities of the negation of the conjecture with rules in $\mathcal{R}$.

Example 4.4 As another example, consider a theory of *lists* with *cons, car, cdr* as discussed in [23, 27]. The defining rules for the interpreted function symbols are:

$$\begin{array}{lll} 1. & cons(car(x),\ cdr(x)) & \rightarrow\ x, \\ 2. & car(cons(x,y)) & \rightarrow\ x, \\ 3. & cdr(cons(x,y)) & \rightarrow\ y. \end{array}$$

These three rules can be shown to constitute an admissible rewrite system. The formula $F \equiv cons(car(x), cdr(car(y))) = cdr(cons(y,x)) \supset cdr(car(y)) = cdr(x)$ can be shown to follow from the rules as follows.

$$4.\quad cons(car(a), cdr(car(b))) \ \rightarrow\ cdr(cons(b,a)).$$

Rule 4 superposes with rule 3 to produce:

$$5.\quad cdr(car(b)) \ \rightarrow\ cdr(a),$$

from which the contradiction follows.

This complete decision procedure for the quantifier-free theory of an admissible rewrite system $\mathcal{R}$ can be combined with the complete decision procedure for Presburger arithmetic with uninterpreted symbols to give a complete decision procedure in the case of both uninterpreted and interpreted symbols. Similar to the case of Presburger arithmetic with uninterpreted symbols, functional terms are abstracted in a conjecture using new abstraction constants. These ground equalities and other ground equalities in a conjecture are then completed using ground completion; additional equalities are generated by superposition of ground equalities with $\mathcal{R}$. New equalities are used to normalize linear inequalities, from which additional implicit equalities are deduced. This interplay similar in the case of uninterpreted symbols, gives a complete decision procedure when interpreted symbols are axiomatized by an admissible rewrite system.

There exist equational theories which have a decidable quantifier-free theory but they do not admit admissible rewrite systems. Furthermore, there are equational theories with a decidable quantifier-free theory which have a canonical rewrite system, but the rewrite system is not admissible.

5 Fourier's Algorithm and Conditional Rewriting

Functions on commonly used data structures such as lists, sequences, arrays, records, etc., are typically expressed using conditional rewrite rules; unconditional rewrite rules are not sufficient. Further lemmas and theorems about functions are also typically conditional rewrite rules. Tecton and its theorem prover *RRL* support definitions and lemmas given as conditional rewrite rules.

Definition 5.1 A *conditional rewrite rule* is a rule of the form

$$lhs \rightarrow rhs \ \ if \ \ \ p_1 \wedge p_2 \cdots \wedge p_k,$$

where lhs is the left-hand side of the rewrite rule, rhs is the right-hand side and $p_1, p_2, \ldots, p_k$ are the *conditions*.

A term $t[t']$, where t' is a subterm of t, rewrites to another term s using a conditional rewrite rule $l \rightarrow r \ \ if \ \ p_1 \wedge \ldots \wedge p_n$, if there is a substitution γ such that $\gamma(l) = t'$ and each of the conditions $\gamma(p_i)$ reduces to *true* recursively also by rewriting using rules; s is t with the subterm t' replaced by $\gamma(r)$. If $\mathcal{R}$ does not include such a rule, then t is said to be in a normal form. As should be evident, conditional rewriting is a recursive process since in order to apply a rewrite rule, its conditions must reduce to true, which itself is determined by rewriting. Termination of this process is guaranteed by using a termination ordering $>_t$ in which $l >_t r$ as well as $l >_t p_i$. For a detailed treatment of conditional rewriting used in *RRL* , see [29, 30].

For combining a complete decision procedure for Presburger arithmetic with a decision procedure for interpreted symbols given as a canonical (conditional) rewrite system, it is possible to identify conditions on a conditional rewrite system similar to the one discussed for unconditional rewrite rules in the previous section. The effectiveness of such an approach is unclear because firstly, it may not be possible to generate a canonical conditional rewrite system, and then secondly, generating additional ground equalities from conditional rewrite rules and equalities can be expensive. In Tecton, we have attempted to make a compromise. We do not assume that interpreted symbols are defined using a canonical conditional rewrite system. We also do not perform any superpositions between the conditional rewrite system and the ground equalities obtained from the conditions of the conclusion. So the implementation is incomplete, but our experience in using it suggests that it works in most situations. Before giving all the steps performed in our preliminary implementation, we will use a streamlined example to illustrate the main features of the implementation.

Example 5.2 A goal to prove is:

$$(p(x) \wedge (z \leq f(max(x,y))) \wedge (0 < min(x,y)) \wedge (x \leq max(x,y)) \wedge (max(x,y) \leq x)) \supset (z < g(x) + y)$$

Among the rewrite rules in the rewrite system are the following two rewrite rules for the interpreted symbols max, f, g, p.

$$\begin{array}{ll} R_1: & min(x,y) \rightarrow y \quad if \quad max(x,y) = x \\ R_2: & f(x) \leq g(x) \rightarrow true \quad if \quad p(x) \end{array}$$

The goal is negated and Skolemized to give:

$$\begin{array}{l} p(A) \wedge (L \leq f(max(A,B))) \wedge (0 < min(A,B)) \\ \wedge (A \leq max(A,B)) \wedge (max(A,B) \leq A)) \wedge \neg(L < g(A) + B). \end{array}$$

(To save space, we do not use abstraction constants to abstract functional terms.) The literal $p(A)$ is the only one not belonging to Presburger language. The set of linear inequalities are:

$$\begin{array}{llllll} C_1: & -f(max(A,B)) + L \leq 0, & C_2: & -min(A,B) \leq -1, & C_3: & -max(A,B) + A \leq 0, \\ C_4: & max(A,B) - A \leq 0, & C_5: & g(A) + B - L \leq 0 & & . \end{array}$$

Because $C_3 + C4 = 0 \leq 0$, the implicit equality

$$E_1: \quad max(A,B) = A$$

is derived by Fourier's algorithm. This equality is used to simplify the inequality set to produce the following inequality set

$$C_1': \quad -f(A) + L \leq 0, \quad C_2: \quad -min(A,B) \leq -1, \quad C_5: \quad g(A) + B - L \leq 0.$$

Now L can be eliminated to give:

$$C_2: \quad -min(A,B) \leq -1, \quad C_5': \quad -f(A) + g(A) + B \leq 0.$$

Since $min(A,B)$ matches the left side of rule R_1 using the substitution $\{x \leftarrow A,\ y \leftarrow B\}$, its condition $max(A,B) = A$ must be established. This equality is already known, so $min(A,B)$ is reduced to B; the equality $min(A,B) = B$ is also added to the equality set. Assuming an ordering $f > g$, term $f(A)$ also matches a maximal term in the linear rule R_2, so the condition $p(A)$ must be established, which is already in the equality set. So, the instance of the linear rule, $f(A) \leq g(A)$, is added to the inequality set, giving:

$$C_2': \quad -B \leq -1, \quad C_5': \quad -f(A) + g(A) + B \leq 0, \quad C_6: \quad f(A) - g(A) \leq 0.$$

Fourier's algorithm detects a contradiction because an inequality $0 \leq -1$ is generated, implying that the original goal is proved.

5.1 Implementation

In this subsection, we give a brief sketch of our preliminary implementation. Many design decisions were greatly influenced by the discussion in Boyer and Moore's report [3] in which they extensively discussed data structures for representing a data base of contexts and the interaction between the rewriter and the decision procedure.

Our preliminary implementation works as follows. Given a goal G, it is negated and Skolemized. The negated Skolemized goal may be divided into many subgoals by splitting at the top most level if there is a disjunction. Each of the subgoals which is of the form $L_1 \wedge \cdots \wedge L_k$ where each L_i is a literal, is attempted for unsatisfiability. From L_i's, a set GE of ground equalities (including literals such as $p(a)$ or $p(a) = false$) is collected. Below, we give the steps:

1. Ground completion is performed to generate a canonical rewrite system for ground equalities. If a contradiction is detected, the subgoal is unsatisfiable. Otherwise, the canonical rewrite system is used to normalize inequalities.

2. The normalized inequalities are passed to Fourier's algorithm for elimination. (a) Whenever an implicit equality is generated, it is passed to step 1, which is repeated. (b) If a contradiction is detected, then the subgoal is unsatisfiable. After steps 1 and 2, if no progress can be made, go to the next step.

3. A maximal literal (term) among all the ground inequalities and ground equalities generated is selected. This literal is checked for matchability with the left side of a rule defining an interpreted symbol. In the case of a linear rule (such as R_2 above relating f, g, p), a maximal term in the left side of a rule is used for checking a possible match.

   Suppose θ is a substitution under which a match is possible with a rule; it is then checked whether the condition in the rewrite rule, if any, instantiated by θ follows from the remaining literals (ground equalities and inequalities generated so far) and other rewrite rules defining interpreted symbols appearing in the condition. This is done using contextual rewriting [29, 30]. If the condition in the rewrite rule is established, then the subterm in the goal matching the left side is replaced by its right hand side instantiated with θ. The above steps are then repeated on the result.

   This process of checking for matchability is itself recursive, as it may involve doing steps 1, 2, and 3 on conditions (on a smaller input in a well-founded order).

   If at any step, it is not possible to reduce a maximal literal, then the procedure terminates declaring that it is unable to prove the subgoal and hence, the original goal.

6 Further Extensions

We have used the Tecton proof system to verify properties of many problems, including sorting algorithms such as insertion sort and quicksort, string matching algorithms, the termination of Takeuchi's function [22], many efficient iterative programs for computing arithmetic functions using the so-called Russian peasant algorithm. Recently, we have also verified properties of parallel programs expressed using a powerlist data structure and recursion as proposed by Misra [21]. The use of the procedure for Presburger arithmetic has made the proofs compact and relatively easier to automate and understand in contrast to proofs generated without using Presburger arithmetic.

In our initial implementation, Fourier's algorithm detected inconsistency if an inequality $0 \leq c$ is derived where $c < 0$. No attempt was made to check for unsatisfiability of integral equalities, or to check whether a satisfying assignment could be generated from defining constraints on the variables eliminated. In the case of natural numbers, we limited the amount of case analysis by considering only some of the conditions generated from Peano's subtraction operation. This was again based on efficiency considerations. We are currently extending our implementation to include additional checks discussed in this paper. Further, some aspects of computing superposition of ground equalities with conditional rewrite rules defining interpreted function symbols that can be performed efficiently, would also be included, and this is likely to extend the class of formulas that the Tecton system would be able to handle automatically.

Acknowledgement: We thank Mahadevan Subramaniam for helpful comments on earlier drafts of the paper.

References

[1] W.W. Bledsoe, A new method for proving certain Presburger formulas, *Proc. 4th IJCAI*, Tbilisi, Georgia, 1975, 15-21.

[2] W.W. Bledsoe and R. Shostak, A prover for general inequalities, 6th IJCAI, 1979, 66-69.

[3] R.S. Boyer and J S. Moore, Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic, *Machine Intelligence* 11 (1988) 83–157.

[4] D.C. Cooper, Theorem proving in arithmetic without multiplications, *Machine Intelligence* 6 (1972) 43–59.

[5] D. Craigen, S. Kromodimoelijo, I. Meisels, W. Pase, and M. Saaltink, Eves system description, *Proc. Automated Deduction - CADE 11,* LNAI 607, Springer Verlag (1992), 771-775.

[6] H. Enderton, *A Mathematical Introduction to Logic.* Academic Press, 1977.

[7] L. Hodes, Solving problems by formula manipulation, *Proc. 2nd Intl. Conf. on AI,* The British Computer Society, 1971, 553-559.

[8] G. Huet and D. Lankford, On the uniform halting problem for term rewriting systems, INRIA Report 283, March 1978.

[9] J. Jaffar, M.J. Maher, P.J.Stuckey, and R.H.C. Yap, "Beyond finite domains," Proc. *Second Workshop on Principles and Practices of Constraint Programming* (*PPCP '94*), Orcas Island, Washington, May 1994, 77-84.

[10] D. Kapur, An automated tool for analyzing completeness of equational specifications, Proc. *Intl. Symp. on Software Testing and Analysis,* Seattle, August 1994.

[11] D. Kapur, D.R. Musser, and X. Nie, "An Overview of the Tecton Proof System," accepted for publication in *Theoretical Computer Science* Journal, special issue on *Formal Methods in Databases and Software Engineering,* (ed. V. Alagar), Vol. 133, October, 1994.

[12] D. Kapur and P. Narendran, The Knuth-Bendix completion procedure and Thue systems, *SIAM Journal on Computing* 14(4), (Nov. 1985), 1052-1072.

[13] D. Kapur and X. Nie, Reasoning about Numbers in Tecton. Tech. Report, Dept. of Computer Science, State University of New York, Albany, NY 12222, 1994.

[14] D. Kapur and M. Subramaniam, Using linear arithmetic procedure for generating induction schemes in mechanizing induction, submitted to *FSTTCS, 1994.*

[15] D. Kapur, and H. Zhang, An overview of Rewrite Rule Laboratory (RRL), to appear in a special issue of *Computers in Math. with Applications,* 1994. Earlier descriptions appeared in CADE-88 and RTA-89.

[16] T. Käufl, Reasoning about systems of linear inequalities, *Proceedings of 9th International Conference on Automated Deduction* 563–572, Argonne, Illinois (1988).

[17] D. Knuth, *The art of computer programming: Seminumerical algorithms.* Vol. 2, Second Edition, 4.5.2, 326-328, Addison-Wesley. 1981.

[18] D. Knuth and P. Bendix, Simple word problems in universal algebras, in *Computational Problems in Abstract Algebra* (ed. Leech), Pergamon Press (1970), 263-297.

[19] J.-L. Lassez, T. Huynh and K. McAloon, Simplification and elimination of redundant linear arithmetic constraints, *Proc. North American Conf. on Logic Programming,* 37-51, 1989.

[20] J.-L. Lassez and M.J. Maher, On Fourier's algorithm for linear arithmetic constraints, *J. of Automated Reasoning,* 9, 1992, 373-379.

[21] J. Misra, "Powerlist: A structure for parallel recursion," *A Classical Mind: Essays in Honor of C.A.R. Hoare,* Prentice Hall, Jan. 1994.

[22] J S. Moore, A mechanical proof of the termination of Takeuchi's function. *Information Processing Letters* 9(4): 176 – 181 (1979).

[23] G. Nelson and D.C. Oppen, Simplification by cooperating decision procedures, *ACM Transactions on Programming Languages and Systems* 1 (2) (1979) 245 – 257.

[24] W. Pugh, A practical algorithm for exact array dependence analysis, *Communications of the ACM*, 35(8), (1992), 112-114.

[25] R.E. Shostak, On the SUP–INF method for proving presburger formulas, *Journal of ACM* 24 (4) (1977) 529–543.

[26] R.E. Shostak, A practical decision procedure for arithmetic with function symbols, *Journal of ACM* 26(2) (1979) 351–360.

[27] R.E. Shostak, Deciding combination of theories, *Journal of ACM* 31 (1), (1984) 1-12.

[28] H.P. Williams, Fourier-Motzkin elimination extension to integer programming problems, *Journal of Combinatorial Theory (A)*, 21, (1976), 118-123.

[29] H. Zhang, Reduction, Superposition, and Induction: Automated Reasoning in an Equational Logic. Ph.D. Thesis. Rensselaer Polytechnic Institute, Computer Science Department (1988).

[30] H. Zhang, Implementing contextual rewriting. In: Proc. *Third International Workshop on Conditional Term Rewriting Systems,* J. L. Remy and M. Rusinowitch (eds.), Lecture Notes in Computer Science, Vol. 656, Springer-Verlag, Berlin, 1992, pp. 363–377.

[31] H. Zhang, D. Kapur, and M.S. Krishnamoorthy, "A mechanizable induction principle for equational specifications," *Proc. of Ninth International Conference on Automated Deduction (CADE-9),* Argonne, IL. Springer-Verlag LNCS 310, 250-265, 1988.

REASONING FROM DATA IN THE MATHEMATICAL THEORY OF EVIDENCE

Mieczysław A. Kłopotek
Institute of Computer Science, Polish Academy of Sciences
01-237 Warsaw, ul. Ordona 21, POLAND, e-mail: klopotek@plearn.bitnet

Abstract

Mathematical Theory of Evidence (MTE) is known as a foundation for reasoning when knowledge is expressed at various levels of detail. Though much research effort has been committed to this theory since its foundation, many questions remain open. One of the most important open questions seems to be the relationship between frequencies and the Mathematical Theory of Evidence. The theory is blamed to leave frequencies outside (or aside of) its framework. The seriousness of this accusation is obvious: no experiment may be run to compare the performance of MTE-based models of real world processes against real world data.

In this paper we develop a frequentist model of the MTE bringing to fall the above argument against MTE. We describe, how to interpret data in terms of MTE belief functions, how to reason from data about conditional belief functions, how to generate a random sample out of a MTE model, how to derive MTE model from data and how to compare results of reasoning in MTE model and reasoning from data.

It is claimed in this paper that MTE is suitable to model some types of destructive processes

Keywords: Approximate Reasoning, Learning System, Modeling Destructive Processes

1. INTRODUCTION

The Dempster-Shafer Theory or the Mathematical Theory of Evidence (MTE) [1, 2] shows one of possible ways of application of mathematical probability for subjective evaluation and is intended to be a generalization of bayesian theory of subjective probability [3].

This theory offers a number of methodological advantages like: capability of representing ignorance in a simple and direct way, compatibility with the classical probability theory, compatibility of boolean logic and feasible computational complexity [4].

MTE may be applied for (1) representation of incomplete knowledge, (2) belief updating, (3) and for combination of evidence [5].MTE covers the statistics of random sets and may be applied for representation of incomplete statistical knowledge. Random set statistics is quite popular in analysis of opinion polls whenever partial indecisiveness of respondents is allowed [6].

Practical applications of MTE include: integration of knowledge from heterogeneous sources for object identification [7], technical diagnosis under unreliable measuring devices [8], medical applications: [9, 10].

An important example concerning difference between implications of bayesian reasoning and reasoning within MTE can be found in [11].

In spite of indicated merits, MTE experienced sharp criticism from many sides. The basic line of criticism is connected with the relationship between the belief function (the basic concept of MTE) and frequencies. A number of attempts to interpret belief functions in terms of probabilities have failed so far to produce a fully compatible interpretation with MTE - see e.g. [12, 13, 14] etc. Shafer [3] and Smets [15], in defense of MTE, dismissed every attempt to interpret MTE frequentistically. Shafer stressed that even modern (that meant bayesian) statistics is not frequentistic at all (bayesian theory assigns subjective probabilities), hence frequencies be no matter at all. Smets stated that domains of MTE applications are those where "we are ignorant of the existence of probabilities", and warns that MTE is "not a model for poorly known probabilities" ([15], p.324). Smets states further "Far too often, authors concentrate on the static component (how beliefs are allocated?) and discover many relations between TBM (transferable belief model of Smets) and ULP (upper lower probability) models, inner and outer measures (Fagin and Halpern [16]), random sets (Nguyen [17]), probabilities of provability (Pearl [18]), probabilities of necessity (Ruspini [19]) etc. But these authors usually do not explain or justify the dynamic component (how are beliefs updated?), that is, how updating (conditioning) is to be handled (except in some cases by defining conditioning as a special case of combination). So I (that is Smets) feel that these partial comparisons are incomplete, especially as all these interpretations lead to different updating rules." ([15], pp. 324-325). Ironically, Smets gives later in the same paper an example of belief function ("hostile-Mother-Nature-Example") which may be clearly considered as lower probability interpretation of belief function, just, at further consideration, leading to very same pitfalls as approaches criticized himself.

Wasserman [20] strongly opposed claims of Shafer [3] about frequencies and bayesian theory. Wasserman pointed out that the major success story of bayesian theory is the exchangeability theory of de Finetti, which treats frequency based probabilities as a special case of bayesian belief. Hence frequencies, as Wasserman claims, are inside the bayesian theory, but outside the Mathematical Theory of Evidence.

This paper is intended to shed some light onto the dispute on relationship between MTE and frequencies. Section 2 introduces basic definitions of MTE. Section 3 clarifies the problem under consideration. Section 4 describes, how to interpret data in terms of MTE belief functions. Section 5 describes a proposal of a model for structuring n of MTE belief functions. belief functions. Section 6 shows how to generate a random sample out of a MTE model of reality. Section 7 describes how to derive MTE model from data. Section 8 is devoted to comparison of results of reasoning in MTE model and reasoning from data. Section 9 indicates open research problem not considered so far within the presented interpretational framework of Mathematical Theory of Evidence.

2. BASIC DEFINITIONS OF MTE

Let us first remind basic definitions of MTE:

Definition 1 *Let Ξ be a finite set of elements called elementary events. Any subset of Ξ be a composite event. Ξ be called also the frame of discernment.*
A basic probability assignment function is any function $m{:}2^{\Xi} \rightarrow [0,1]$ such that

$$\sum_{A\in 2^{\Xi}} |m(A)| = 1, \quad m(\emptyset) = 0, \quad \forall_{A\in 2^{\Xi}} \quad 0 \leq \sum_{A\subseteq B} m(B)$$

(|.| - absolute value.

A belief function be defined as $Bel{:}2^{\Xi} \rightarrow [0,1]$ so that

$$Bel(A) = \sum_{B\subseteq A} m(B)$$

A plausibility function be $Pl{:}2^{\Xi} \rightarrow [0,1]$ with

$$\forall_{A\in 2^{\Xi}} \; Pl(A) = 1 - Bel(\Xi - A)$$

A commonality function be $Q{:}2^{\Xi} - \{\emptyset\} \rightarrow [0,1]$ with

$$\forall_{A\in 2^{\Xi}-\{\emptyset\}} \quad Q(A) = \sum_{A\subseteq B} m(B)$$

Furthermore, a Rule of Combination of two Independent Belief Functions Bel_1, Bel_2 Over the Same Frame of Discernment (the so-called Dempster-Rule), denoted

$$Bel_{E_1,E_2} = Bel_{E_1} \oplus Bel_{E_2}$$

is defined as follows: :

$$m_{E_1,E_2}(A) = c \cdot \sum_{B,C;A=B\cap C} m_{E_1}(B) \cdot m_{E_2}(C)$$

(c - constant normalizing the sum of $|m|$ to 1)

Furthermore, let the frame of discernment Ξ be structured in that it is identical to cross product of domains Ξ_1, Ξ_2, $\ldots \Xi_n$ of n discrete variables $X_1, X_2, \ldots X_n$, which span the space Ξ. Let $(x_1, x_2, \ldots x_n)$ be a vector in the space spanned by the variables $X_1, . X_2, \ldots X_n$. Its projection onto the subspace spanned by variables $X_{j_1}, X_{j_2}, \ldots X_{j_k}$ ($j_1, j_2, \ldots j_k$ distinct indices from the set 1,2,. . . ,n) is then the vector $(x_{j_1}, x_{j_2}, \ldots x_{j_k})$. $(x_1, x_2, \ldots x_n)$ is also called an extension of $(x_{j_1}, x_{j_2}, \ldots x_{j_k})$. A projection of a set A of such vectors is the set $A^{\downarrow X_{j1}, X_{j2}, \ldots X_{jk}}$ of projections of all individual vectors from A onto $X_{j_1}, . X_{j_2}, \ldots X_{j_k}$. A is also called an extension of $A^{\downarrow X_{j1}, X_{j2}, \ldots X_{jk}}$. A is called the vacuous extension of $A^{\downarrow X_{j1}, X_{j2}, \ldots X_{jk}}$ iff A contains all possible extensions of each individual vector in $A^{\downarrow X_{j1}, X_{j2}, \ldots X_{jk}}$. The fact, that A is a vacuous extension of B onto space $X_1, X_2, \ldots X_n$ is denoted by $A = B^{\uparrow X_1, X_2, \ldots X_n}$

Definition 2 *Let m be a basic probability assignment function on the space of discernment spanned by variables $X_1, X_2, \ldots X_n$. $m^{\downarrow X_{j1}, X_{j2}, \ldots X_{jk}}$ is called the projection of m onto subspace spanned by $X_{j_1}, X_{j_2}, \ldots X_{j_k}$ iff*

$$m^{\downarrow X_{j1}, X_{j2}, \ldots X_{jk}}(B) = c \cdot \sum_{A; B = A^{\downarrow X_{j1}, X_{j2}, \ldots X_{jk}}} m(A)$$

(c - normalizing factor)

Definition 3 *Let m be a basic probability assignment function on the space of discernment spanned by variables $X_{j_1}, X_{j_2}, \ldots X_{j_k}$. $m^{\uparrow X_1, X_2, \ldots X_n}$ is called the vacuous extension of m onto superspace spanned by $X_1, X_2, \ldots X_n$ iff*

$$m^{\uparrow X_1, X_2, \ldots X_n}(B^{\uparrow X_1, X_2, \ldots X_n}) = m(B)$$

and $m^{\uparrow X_1, X_2, \ldots X_n}(A) = 0$ for any other A.
We say that a belief function is vacuous iff $m(\Xi) = 1$ and $m(A) = 0$ for any A different from Ξ.

Projections and vacuous extensions of Bel, Pl and Q functions are defined with respect to operations on m function. Notice that by convention if we want to combine by Dempster rule two belief functions not sharing the frame of discernment, we look for the closest common vacuous extension of their frames of discernment without explicitly notifying it.

Definition 4 *(See [21]) Let B be a subset of Ξ, called evidence, m_B be a basic probability assignment such that $m_B(B) = 1$ and $m_B(A) = 0$ for any A different from B. Then the conditional belief function $Bel(.||B)$ representing the belief function Bel conditioned on evidence B is defined as: $Bel(.||B) = Bel \oplus Bel_B$.*

3. PROBLEM STATEMENT

We notice easily, that if the function m is positive only for sets with cardinality 1 then Bel is the plain finite discrete probability function. On relationship between probability functions and bayesian rule consult e.g. [13]. As m sums up to 1 on the set of sets of elementary events we may be tempted to consider m as an ordinary probability function. But, as cited previously from the work of Smets, this view contradicts the Dempster rule of combination of independent Bels.

In fact, probability functions give rise to some expectations, which cannot be satisfied by a belief function. Generally, we feel that a theory of real world is correct if we make initial observations, let run the intrinsic real world process and the theoretical process on this set of observations and then results of both processes coincide. Probability theory matches this expectation. If we model the reality with a joint probability distribution P() and set a selection condition to membership in set B, start a real world process generating independent events following distribution P() and out of these events we strictly select those fitting condition B, and then within those selected events we estimate a joint probability distribution P'(), then P'() and P() are related (in the limit) by the theoretical model of conditional distribution $P'(A) = P(A|B)$. The major trouble with belief functions is that no such frequentist interpretation exists for them. Models criticized by Smets [15] and many other fail to define such an interpretation of initial data, the process and the results as to fit Dempster rule of independent evidence combination.

Smets and Shafer propose a simple solution to this problem: do not look for such real world processes at all. MTE shall exist as an nicely shaped abstract object in the space of philosophical thinking devoid of any practical meaning.

We propose here another way out. We shall insist on looking for processes meeting MTE requirements and rethink the type of process governing probabilistic reasoning.

4. A FREQUENTISTIC INTERPRETATION OF BELIEF FUNCTIONS

A detailed presentation of our interpretation is presented elsewhere [22].

Essentially, we assume (like in papers of Nguyen [17]) that each object of the population has set-valued attribute(s). But unlike other approaches we assume that objects are attached labels. Labels are subsets of the frame of discernment Ξ and indicate which values are permissible for consideration for a given object. Then the belief function Bel(A) measures for the set A the share of objects for which the intersection of the attribute value of the object and of the label of this object (and NOT solely the attribute value) is contained in A (m, Pl, and Q are easily derived from Bel).

The idea of a label reflects subjectiveness within the framework of MTE. It happens in the real life that community attaches vicious labels to a man remembering only his weaknesses (which he surely has) and totally ignoring his virtues (which he may possibly have). So the label of an MTE object may express a kind of irreversible prejudice, or attitude, or belief of a community with respect to this object. We can also imagine a process of medical diagnosis. An iniatal set of hypotheses is proposed. Then various medical tests are run which label the patient with different sets of hypotheses, and the intersection of those labels and the initial finding is considered to be the actual ilness (though no single test may be possible to distinguish this single ilness from all the other). Another type of labeling may be that of actual loosing some properties an object had once upon a time. Someone qualified today as a contemporary example good father, may loose this property in - say - 40 years (He may die by then or be just a good grandfather). A match may loose its feature as being able to light a fire etc.

The importance of a label for the MTE object is operational. Let us assume we run a deterministic MTE "selection" process $proc_B$ as follows: if the intersection of B, the object's attribute value and its label is empty then we reject the object from the population (negative selection, identical with a probabilistic conditioning process). Otherwise we change the label of the object to intersection of B and its pre-process label (this is unlike probabilistic conditioning where we leave accepted objects unchanged). Let us denote by Bel' the belief function derived from the population obtained after the process Bel_B. It is easily seen that then $Bel' = Bel \oplus Bel_B = Bel(.||B)$.

We may also consider a non-deterministic family of such labeling processes $proc_{B_1}$, $proc_{B_2}. \dots proc_{B_k}$ where independently for each object randomly one of the processes is chosen with probability $Pr(B_1)$, $Pr(B_2)$, $\dots Pr(B_k)$ respectively. Let Bel denote the belief function from frequencies in the initial population, Bel" the belief function after the run of combined processes, and Bel_{proc} shall denote such a belief function that for j=1,...,k $m_{proc}(B_k) = Pr(B_k)$. Then again is is easily demonstrated that the expected value of Bel" is equal $Bel" = Bel \oplus Bel_{proc}$.

Let us illustrate the idea with the following example.

Let us assume that Mr.AX, Mr.BY, Mr.CZ and Mr.DT are major contributors of (imaginary) Formal Theory of Formality (FTF). We shall evaluate their relative contribution to the theory - just by measuring the number of publications. We collected 100 papers some of which were written by a single author, some by several of them. The overall statistics is presented in the subsequent table:

Author(s)	Number of publications
AX	5
BY	15
CZ	8
DT	2
AX,BY	24
AX,CZ	11
AX,BY,CZ	20
AX,BY,DT	9
AX,BY,CZ,DT	6
TOTAL	100

fun/arg	$\{AX\}$	$\{BY\}$	$\{AX, BY\}$
m	0.05	0.15	0.24
Bel	0.05	0.15	0.44
Pl	0.75	0.74	0.90
Q	0.75	0.74	0.59

The neighboring table shows the values of MTE measures for three sets of authors consisting of : (1) only AX, (2)only BY and (3) both AX and BY.

Let us assume that we obtained a "hint" from a "friendly person" that Mr.DT is in fact a fictive author of FTF. This information ("evidence") may be captured by the belief function Bel_1 such that $m_1(\{AX, BY, CZ\}) = 1$. How to use this hint

? We may take the papers we collected, one by one, and delete Mr.DZ from the list of authors of the paper, and if no author is left, we throw the paper away. After such an operation we obtain the new "statistics" of contributions:

Author(s)	Number of papers:
AX	5
BY	15
CZ	8
AX,BY	33
AX,CZ	11
AX,BY,CZ	26
TOTAL	98

fun/arg	$\{AX\}$	$\{BY\}$	$\{AX, BY\}$
m'	5/98	15/98	33/98
Bel'	5/98	15/98	53/98
Pl'	75/98	74/98	90/98
Q'	75/98	74/98	59/98

Bel' be the new belief function for the new population. It is easily seen that $Bel' = Bel \oplus Bel_1$.

Let another, independent friendly person give us another hint that also CZ is a fictive contributor to FTF. But let us trust this person only to 70 %. This fact should be represented by the belief function Bel_2 such that $m_2(\{AX, BY, DT\}) = 0.7$ and $m_2(\{AX, BY, CZ, DT\}) = 0.3$.

Let us consume this new "evidence" as follows: we take papers (after last operation) one by one, throw a properly biased coin (70% heads, 30 % tails), toss it and on heads we delete CZ from the list of authors (and throw the paper away if no other author is left) and on tails we accept the paper as is. The expected value over "statistics" of a process like this is visible below.

Author(s)	Number of papers:
AX	8.3
BY	15
CZ	5.6
AX,BY	40.8
AX,CZ	7.7
AX,BY,CZ	18.2
TOTAL	95.6

Let *Bel"* denote the belief function after the second process. Provably its expected value may be calculated as $Bel" = Bel' \oplus Bel_2$.

5. MTE MODELS

It is usually (next to) impossible to represent directly a (joint) probability distribution in a larger number of variables. E.g. 15 four-valued variables would require 1 Giga floating point cells. With MTE belief functions it is even worse. A direct representation of a belief function in 15 variables with four-valued frame of discernment each would require 1 billion Giga floating point cells. Therefore in both domains alternative representations are looked for. Within probability domain so-called bayesian networks are used [23], exploiting conditional independences, where the joint probability distribution is represented as a combination of conditional probabilities of variables X_i on their direct causes $X_{\pi(i)}$:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_{\pi(i)})$$

Within the MTE literature another kind of "belief networks" is used, due i.e. to Shenoy and Shafer [24]. It is so-called factorization along a hypergraph:

$$Bel = Bel_1 \oplus Bel_2 \oplus \ldots \oplus Bel_m$$

where no requirements are put on the form of components Bel_i. This approach has been criticized recently by Cano et al. [25] as - contrary to Pearl's bayesian networks - these MTE belief networks don't reflect (conditional) independences among variables. Shafer's conditional belief function cannot be candidate for a factor in belief function factorization. Therefore they proposed a definition of (so-called apriorical) conditionality within MTE saying that Bel is a conditional belief function conditioned on variables $X_1, \ldots X_k$ iff $Bel^{\downarrow X_1,\ldots,X_k}$ is a vacuous belief function. They require the factorization of a belief function to be in terms of such conditional belief functions. However, this definition is counterproductive as for the special case of belief function being a bayesian distribution the whole bayesian network may collapse into a single node of Cano's belief network.

We proposed therefore another definition of apriorical conditional belief function in [26] and demonstrated there that Shenoy/Shafer factorization cannot be simpler than that into a belief network of ours.Apriorical conditional belief function $Bel^{|X_1,\ldots,X_k}$ reflecting conditioning of Bel on $X1, \ldots, X_k$ is any pseudo-belief function satisfying

$$Bel = Bel^{|X_1,\ldots,X_k} \oplus Bel^{\downarrow X_1,\ldots,X_k}$$

(A pseudo-belief functions allows m's to be negative, but only so that respective Q's remain non-negative). We then define a belief network factorization of Bel with frame of discernment spanned by variables $X_1, \ldots, X_n$ as

$$Bel = \bigoplus_{i=1}^{n} Bel^{\downarrow \{X_i\} \cup X_{\pi(i)} | X_{\pi(i)}}$$

6. GENERATING RANDOM SAMPLES FROM MTE MODELS

Generating a random sample from the bayesian network is a relatively simple task. We have to consider an ordering of variables compatible with the partial order imposed by the network. To generate a single object, we take variable by variable following this ordering, look at values $x_{\pi(i)}$ already assigned to direct causes $X_{\pi(i)}$ of

X_i and then we select randomly one of the values $x_{i1}, x_{i2}, \ldots, x_{in_i}$ taken by variable X_i with probabilities proportional to $P(x_{i1}|x_{\pi(i)}), P(x_{i2}|x_{\pi(i)}), \ldots, P(x_{in_i}|x_{\pi(i)})$ That is, in a single pass we can generate one sample object. (All random selections should be independent of one another).

It is not that easy with belief function models in general. To generate a single sample object we have to assign it first the frame of discernment set Ξ, calling this set O_0. Then we take one after the other component Bel_i function (ordering may be anyhow), eventually make the vacuous extension $Bel_i^{\uparrow}$ of it onto the frame of discernment. Then we select randomly a set $A_j \subseteq \Xi$ according to probability distribution proportional to $|m_i^{\uparrow}(A_j)|$. If at the time the object had the value O_{i-1}, then we assign it a new value $O_i = O_{i-1} \cap A_j$. Should O_i be an empty set, then we terminate the pass without generating an object (a failure). That is, within a single pass we may or may not generate a sample object.

With pseudobelief functions the situation is even worse. The object is within each step assigned a set and a mark + ore -. If the $m_i(A_j)$ of selected A_j is positive, then the mark is left unchanged, if it is negative, then the mark is inverted. If within the sample there are two identical objects with respect to the finally assigned set, but with opposed marks, then they are canceled out. After completion of sample generation process all remaining objects with marks '-' are canceled.

However, if the model is in terms of a belief network as described in [26] (see section 5), then the problem of sample generation reduces to probabilistic case (one sample object per single pass), however one has to keep track of '+'/'-' marks as indicated above.

7. EXTRACTING MTE MODELS FROM DATA

One of major advantages of the belief network model proposed by us for MTE is its capability to connect statistical independence from data with d-separation within the underlying directed acyclic graph of the belief network (see [27] for more on this point). d-separation property [23] means possibility of derivation of conditional independence from graphical structure of dependencies. Several algorithms have been proposed for derivation of belief network structure for MTE (see [22, 26, 27]).

Within the framework of DS-JACEK system [28] still another algorithm has been implemented. It is a version of CI algorithm of Spirtes et al. (see [29]. CI algorithm is known of capability to derive causal models under causal insufficiency. It has been adopted for generation of bayesian networks [30], and later for MTE belief networks [31]. The essential adaptation step for MTE is substitution of bayesian conditional independence test with MTE apriorical conditional independence test. If we test conditional independence of variables X and Y on the set of variables Z, then we have to compare empirical distribution $Bel^{\downarrow X,Y,Z}$ with $Bel^{\downarrow X,Z|Z} \oplus Bel^{\downarrow Y,Z|Z} \oplus Bel^{\downarrow Z}$. The traditional χ^2 statistics is computed (treating the latter distribution as expected one). If the hypothesis of equality is rejected on significance level $\alpha = 0.05$ then X and Y are considered dependent, otherwise independent.

8. REASONING IN SHENOY-SHAFER AXIOMATIC FRAMEWORK VERSUS REASONING FROM DATA

The crucial point of the whole effort of developing data model for MTE was to provide a reference point for MTE reasoning engines.

An MTE reasoning engine calculates (Shafer's) conditional marginal distributions from an MTE model and a set of observations. Several such engines have been developed (see [24] for further references, see also [25]). Shenoy/Shafer method of local computations for MTE, described e.g. in [24] works with hypergraph factorization models of belief functions. It has been implemented in the reasoning engine of DS-JACEK [28].

Conditioning for probabilistic databases is achieved in a simple manner. It is sufficient to impose a filter (filtering expression, or select expression) and the calculation of marginals can be done on the physically same dataset just evaluating truth value of filtering expression.

It is not that simple for MTE databases. Neither are MTE constraints deterministic (they are usually non-deterministic) nor is conditioning just the matter of selection. One has to prepare the physical copy of the database. Then take each record case-by-case and apply the same procedure as when generating a random sample with two differences: we do not start with assignment of Ξ to the object, but rather start with its currently assigned set of values. Then we do not use Bels describing the model but rather the ones describing constraints (observations). So the record may be either discarded from the database or retained but changed (Due to possibilities of changes we just require that a physical copy of the original database is to be prepared for the process). Then marginals over the modified copy are just results of MTE reasoning from data.

It is a very important issue to possess two different mechanisms for MTE reasoning: one from model and one from data. We may have diverse testing conditions where we may utilize them in suitable way. On the one hand we may have a model of reality available and want to develop/test a new/old MTE reasoning engine. It is usually to hard to verify results of MTE reasoning for models with more than 10 variables by hand. And usually the implementation of sample generator in combination with sample constraint imposer is much simpler than that of a fine (time and/or space saving) model-based MTE reasoning engine. So we can generate a random sample from the model, run reasoning both via sample constraint imposer and via the developed knowledge-based reasoning engine and then compare resulting (aposterioric) conditional belief functions from data and from the model.

The other setting may be that we have data available and want to develop the proper model of reality. We may generate model from data either by automatic learning program, or in interactive manner or by "enlightened guess". Then we can compare under various constrains the results of reasoning from this model and the data and eventually reject or adjust the model, change/improve model generation algorithms etc..

Last not least, we may have been provided with a model once upon the time and it was not until later that we collected a sufficient amount of data to verify its thorough or partial validity.

9. DISCUSSION AND CONCLUDING REMARKS

The frequentistic interpretation of Mathematical Theory of Evidence described in this paper has been fully implemented: major part of it within a knowledge-base development-and-test system DS-JACEK and partially within the test-bed for this system. For sparsely connected networks of up to 15 dempsterian variables, databases with up to 5,000 cases were randomly generated and thereafter in most cases successfully reconstructed using methodology described above. Cases of failure turned out usually to provide alternative but functionally equivalent structure. Run-times on an PC i486, 66MHz in extreme cases didn't exceed 10 minutes.

Generation of random dempsterian samples served also as a test-bed for the reasoning machine of DS-JACEK. Following of test examples with more than 10 variables by hand may prove not feasible. Hence for larger MTE models random samples of 10,000 cases and more were generated and then results of knowledge-based DS-JACEK reasoning machine were compared with marginals from a sample-based test-bed reasoning machine. Comparisons of this form proved to be quite useful for program development.

With this presentation main application problems of MTE seem to be overcome. Processes changing data in a coherent way (reducing the space of possibilities for each individual) like those of stepwise medical diagnosis, or destructive processes have been identified as a class of processes which can be modeled using Mathematical Theory of Evidence. Belief functions may be calculated from data, verified against data and may serve as source for random generation of data. Independence of dempsterian-shaferian variables acquired statistical meaning. A new category of MTE belief networks has been defined paralleling bayesian belief networks in their capability to capture qualitatively independence and conditional independence in form of a directed acyclic graphs. Algorithms to decompose a data grounded belief function into a belief network have been elaborated.

A number of questions are still left open. :

- how to interpret (if at all) pseudo-belief functions (that is with negative masses in basic probability assignment),
- how to obtain unbiased estimates of apriorical conditional belief functions weighing belief network nodes,
- how to accomplish statistically optimal approximations of belief functions by belief networks from noisy data,
- what should be optimal independence and conditional independence tests for belief network models.

These questions along with related issues are subject of further research. It is due to the work presented in this paper that questions of statistical estimations within the MTE may be posed at all.

Previous work, done among other by Shafer [21] by Kyburg [12], Halpern and Fagin [13], Fagin and Halpern [16. 14], Pearl [33], Grzymala-Busse [34], Nguyen [17], as well as critical evaluations of these approaches e.g. by Smets [15], contributed to clarification of similarities and dissimilarities between frequentistic approaches to probability and to MTE. The major contribution of this paper is to pinpoint the essential difference between probabilistic reasoning and the reasoning within MTE. Probabilistic reasoning can be understood - in terms of objects of the population - as a selection of objects with some properties. Whereas MTE reasoning means selection of objects combined with destruction of some of their properties. So, objects subject to a "probabilistic process" are the same before and after completion of the process. Whereas objects subject to an MTE process are changed, deformed after completion of the process. I think this explains difficulties encountered by alternative frequentistic interpretations of MTE presented in the literature. This fundamental insight allowed for enhancement of MTE with many features reserved so far for probabilistic models only (like representation in terms of a belief network, generation of random samples from belief function models, identification of belief networks from data, comparison of model-based and data-based reasoning etc.).

References

[1] Shafer G.: *A Mathematical Theory of Evidence* , Princeton University Press, Princeton, 1976

[2] Dempster A.P.: Upper and lower probabilities induced by a multi-valued mapping, *Ann. Math. Stat.* 38 (1967), 325-339

[3] Shafer G.: Perspectives on the theory and practice of belief functions, *International Journal of Approximate Reasoning*, 1990:4, 323-362.

[4] Ruspini E.H., Lowrance D.J., Strat T.M.: Understanding evidential reasoning, *International Journal of Approximate Reasoning*, 1992:6, 401-424.

[5] Provan G.M.: The validity of Dempster-Shafer belief functions, *International Journal of Approximate Reasoning*, 1992:6, 389-399.

[6] Dubois D., Prade H.: Evidence, knowledge and belief functions, *International Journal of Approximate Reasoning*, 1992:6, 295-319.

[7] deKorvin A., Kleyle R., Lea R.: The object recognition problem when features fail to be homogenous, *International Journal of Approximate Reasoning* 1993:8, 141-162.

[8] Durham S.D., Smolka J.S., Valtorta M.: Statistical consistency with Dempster's rule on diagnostic trees having uncertain performance parameters, *International Journal of Approximate Reasoning* 1992:6, 67-81.

[9] Gordon J., Shortliffe E.H.: The Dempster-Shafer theory of evidence, in: G. Shafer, J. Pearl eds: *Readings in Uncertain Reasoning*, (ISBN 1-55860-125-2, Morgan Kaufmann Publishers Inc., San Mateo, California, 1990), 529-539.

[10] Zarley D.K., Hsia Y, Shafer G.: Evidential reasoning using DELIEF, *Proc. of the Seventh National conference on Artificial Intelligence (AAAI-88)*, 1, 205-209, Minneapolis, MN, 1988.

[11] Smets Ph., Kennes R.: The tranferable belief model, *Artificial Intelligence 66* (1994), 191-234

[12] Kyburg Jr H.E.: Bayesian and non-Bayesian evidential updating, *Artificial Intelligence* 31 (1987), 271-293.

[13] Halpern J.Y., Fagin R.: Two views of belief: belief as generalized probability and belief as evidence,*Artificial Intelligence* 54(1992), 275-317

[14] Fagin R., Halpern J.Y.: Uncertainty, belief, and probability, *Comput. Intell.* 7(1991), 160-173

[15] Smets Ph.: Resolving misunderstandings about belief functions, *International Journal of Approximate Reasoning* 1992:6:321-344.

[16] Fagin R., Halpern J.Y.: Uncertainty, belief, and probability, *Proc. Int. Joint Conf. AI, IJCAI89*, Detroit, 1161-1167, 1989

[17] Nguyen H.T.: On random sets and belief functions, *J. Math. Anal. Appl.* 65, 539-542, 1978.

[18] Pearl J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Influence*, Morgan and Kaufmann, 1988

[19] Ruspini E.H.: The logical foundation of evidential reasoning, Tech. Note 408, SRI International, Menlo Park, Calif. USA, 1986.

[20] Wasserman L.: Comments on Shafer's "Perspectives on the theory and practice of belief functions", *International Journal of Approximate Reasoning* 1992:6:367-375.

[21] Shafer G., Srivastava R.: *The Bayesian and Belief-Function Formalisms. A General Prospective for Auditing*, in: G. Shafer, J. Pearl eds: *Readings in Uncertain Reasoning*, (ISBN 1-55860-125-2, Morgan Kaufmann Publishers Inc., San Mateo, California, 1990), 482-521.

[22] Kłopotek M.A.: Belief as probability and as update scheme: A reconciled view, submitted

[23] Geiger D., Verma T., Pearl J.: d-Separation: From theorems to algorithms, in : *Uncertainty in Artificial Intelligence 5* (M.Henrion, R.D.Shachter, L.N.Kamal and J.F.Lemmer Eds), Elsevier Science Publishers B.V. (North-Holland), 1990, 139-148.

[24] Shenoy P., Shafer G.: Axioms for probability and belief-function propagation, in: Shachter R.D., Levitt T.S., Kanal L.N., Lemmer J.F. (eds): *Uncertainty in Artificial Intelligence 4*, Elsevier Science Publishers B.V. (North Holland), 1990,

[25] Cano J., Delgado M., Moral S.: An axiomatic framework for propagating uncertainty in directed acyclic networks, *International Journal of Approximate Reasoning*. 1993:8, 253-280.

[26] Kłopotek M.A.: Beliefs in Markov Trees - From Local Computations to Local Valuation, R. Trappl, Ed.: *Cybernetics and Systems Research*, Proc. 12th European Meeting on Cybernetics and System Research 5-8 April 1994, World Scientific Publishers, Vol.1. pp. 351-358

[27] Kłopotek M.A.: Dempsterian-Shaferian Belief Network From Data, submitted

[28] Michalewicz M., Wierzchoń S.T., Kłopotek M.A.: *Knowledge Acquisition, Representation & Manipulation in Decision Support Systems*, [in]: M. Dąbrowski, M. Michalewicz, Z. Raś: *"Intelligent Information Systems"*, Proceedings of a Workshop held in Augustów, Poland, 7-11 June, 1993, ISBN 83-900820-2-0 pp. 210- 238,

[29] Spirtes P., Glymour C., Scheines R.: Causation, Prediction and Search, Lecture Notes in Statistics 81, Springer-Verlag, 1993.

[30] Kłopotek M.A.: Learning Belief Network Structure From Data under Causal Insufficiency, in *Proc. of 13th European Conference on Machine Learning*, Catania Italy, April 1994

[31] Kłopotek M.A.: Handling Causal Insufficiency in Dempster-Shafer Belief Networks-to appear in Proc. Conf. Information Processing and Management of Uncertainty, Paris, 4-8 July 1994

[32] Shafer G.: *Belief functions: An introduction*, in: G. Shafer, J. Pearl eds: *Readings in Uncertain Reasoning*, (ISBN 1-55860-125-2, Morgan Kaufmann Publishers Inc., San Mateo, California, 1990), 473-482.

[33] Pearl J.: Bayesian and Belief-Function formalisms for evidential reasoning:A conceptual analysis, in: G. Shafer, J. Pearl eds: *Readings in Uncertain Reasoning*, (ISBN 1-55860-125-2, Morgan Kaufmann Publishers Inc., San Mateo, California, 1990), 540-569.

[34] Grzymała-Busse J.W.: *Managing Uncertainty in Expert Systems*, Kluwer, 1991.

TABLEAUX OF LOGICS OF PARADOX

Zuoquan Lin*
Computer Science Department
Shantou University, Shantou 515063, CHINA
and
Computer Science Department
BUAA, Beijing 100083, CHINA
Wei Li
Computer Science Department
BAUU, Beijing 100083, CHINA

Abstract

The logic of paradox LP proposed by Priest [1979] is one of paraconsistent logics. One of the motivations behind paraconsistent logic, namely LP, is that it should not be the case that everything follows from a single contradiction. It must pay a price , however, that some classical inferences would be invalid in LP. In Priest's recent invention, the logic of minimal paradox LP_m can overcome the drawback, such that paraconsistent logic would be equivalent to classical logic when there is not direct effect of a contradiction. Although some proof theories for LP were introduced, there has not yet been a satisfactory proof theory for LP_m. We will propose a sound and complete tableaux for LP_m in this article.

1 INTRODUCTION

The *logic of paradox LP* proposed by Priest [1979] is one of well-known paraconsistent logics in the literature (*cf.*, [14]). We say that a theory is *inconsistent* if it contains both A and $\neg A$ for some proposition A. A theory is *trivial* if it contains every proposition. *Paraconsistent logics* are those logics in which an inconsistent theory can be nontrivial. One of the motivations behind paraconsistent logic, namely LP, is that it might get rid of triviality, *i.e.*, it should not be the case that everything follows from a single contradiction. Therefore, paraconsistent logic can be used as a tool for formalizing reasoning in the presence of inconsistency.

*The author is supported in part by National Hi-Tech 863 Project, in part by National Key Project of Fundamental Research Climbing Program and in part by Natural Science Foundation of China.

As a paraconsistent logic, LP can indeed localize contradictions and obtain nontriviality. It must, however, pay a price: some inferences that are classically valid are not valid in LP, so that it would be too weak to permit interesting conclusions [13]. In Priest's recent invention [13], the logic of *minimal* paradox LP_m provided as a *nonmonotonic* extension of LP, among other things, can overcome the drawback. Interestingly, LP_m by introducing nonmonotonicity into LP can overcome the weakness problem of the paraconsistency. LP_m is nonmonotonic in a sense that the inconsistency is *minimal*. This is because the *minimally inconsistent models* of an enlarging set of premises would be changed so that some previous conclusions could be withdrawn when facing with new information .[1] LP_m has some nice properties from which it is considered interesting in formalizing commonsense reasoning with incomplete and inconsistent knowledge [9].

Originally, LP was defined as a semantic entailment of paraconsistent logic. In [7], a reasoning method based on resolution was provided as a satisfactory proof procedure for LP. The method can be easily rewritten as a so-called *mated* tableaux for LP. Essentially the same idea was first used in [3,4] as a proof theory for first-degree entailment of relevance logic [1]. [2] Similarly, a sequent calculus can also be provided for LP as pointed out in [13]. Also, LP_m was originally defined in semantic way. As pointed out in [12], there was an open problem of how to provide a satisfactory proof theory for LP_m. In [8], an attempt to revise mated tableaux for LP to fit LP_m failed as pointed out by the author himself, [3] and the problem was left open and challenge.

In this paper we will provide a *minimal* tableaux as a satisfactory proof theory for LP_m. Firstly, we present another tableaux, *signed tableaux*, for LP by modifying the procedure of *analytic tableaux* in [15]. [4] In spirit, our signed tableaux method is similar to mated tableaux one for LP [7,8] in that classical tableau rules are remained and the closed conditions of tableaux are modified to fit the paraconsistency. Then, we will propose a *minimal tableaux* for LP_m based on signed tableaux for LP. In order to capture the minimality of LP_m by tableaux we must modify the construction of signed tableaux to eliminate the *redundant* branches of of the non-minimally inconsistent models in the tableaux. Technically, our minimal tableaux method is similar to those of the literature [6] and [10] to capture nonmonotonicity, but different in a significant sense to capture paraconsistency. The tableaux are sound and complete with respect to the semantics of LP and LP_m, respectively.

[1] However, the logic LP_m is not well suitable for dealing with the reasoning with incomplete information in the sense of the motivation of nonmonotonic formalism. In [9], the logic of *circumscriptive* paradox was provided as a significant extension of LP_m which remains the nice properties of LP_m and has the ability of circumscription for nonmonotonic reasoning.

[2] There existed some quite tableau systems for other paraconsistent logics differently in spirit, for instance, the calculi C_n introduced by da Costa in the literature [2].

[3] Personal communication, 1992.

[4] Notice that (signed) tableaux has been frequently used as proof procedure for non-classical logics. As we will see in this paper, the method we used is very different from other ones for non-classical logics. It seems that we do not need to list much literatures on tableaux.

The rest of the paper is organized as follows. In Section 2, we review the semantics of LP and LP_m. In Sections 3 and 4, we present the sound and complete tableaux for LP and LP_m, respectively. We make some remarks about related works in the concluding section. The proof of the sound and complete theorems of the tableaux with respect to the semantics of LP and LP_m are presented in the appendix.

2 SEMANTICS OF LP AND LP_m

Throughout this paper, we suppose Las a propositional language, and the (well-formed) formulas are defined as usual. An evaluation π assigns to each atomic proposition p in Lone of the following three values: 0, 1 or 01. We say that p is *true* under π if $\pi(p) = 1$ or $\pi(p) = 01$; and p is *false* under π if $\pi(p) = 0$ or $\pi(p) = 01$. Thus LP is one kind of three-valued semantics in which under an evaluation, some proposition may be *both true and false.*

Under an evaluation, truth values can be extended conventionally to non-atomic propositions as follows: [5]

1. *$\neg A$ is true iff A is false; $\neg A$ is false iff A is true.*

2. *$A \vee B$ is true iff either A or B is true; $A \vee B$ is false iff both A and B is false*

3. *$A \wedge B$ is true iff both A and B are true; $A \wedge B$ is false iff either A or B is fasle.*

And other connectives are defined in the standard way, *e.g.*, $A \rightarrow B$ is defined by $\neg A \vee B$.

We will write $\pi(A) = 1$ if A is *true but not false* under π; $\pi(A) = 0$ if A is *false but not true* under π; and $\pi(A) = 01$ if A is *both true and false* under π. In the following, we say that an evaluation π is a *model* of a formula A (a set S of formulas) if A (every member of S) is *true* under the evaluation .

The *semantic entailment* of LP is defined in the standard way.

Definition 1 *Let S be a set of formulas and A a formula. A is a semantic consequence of S, written as $S \models_{LP} A$, iff A is true in all models of S.*

The properties of logic LP can be easily seen by the following example.

Example 1. Let p and q be two different atomic propositions. It is easy to see that $p \wedge \neg p \models_{LP} p$; but $p \wedge \neg p \not\models_{LP} q$, for under the evaluation π such that $\pi(p) = 01$ and $\pi(q) = 0$, $p \wedge \neg p$ is true (actually it is both true and false) but q is not true.

One of the motivations behind paraconsistent logic is that we should not allow everything to follow from a single contradiction. Thus LP get rid of the

[5] It is easy to illustrate truth tables of the semantics of LP according to the following definition that are actually the same as Kleene's strong three-valued logic.

triviality of classical logic, and as a paraconsistent logic, it indeed in a sense localizes contradictions. It has, however, paid a price: if A is a classically consistent formula and B follows from A in classical logic, then B may not follow from A in LP. Obviously, ***reductio ad absurdum*** fails in LP. Exactly, what fails is the disjunctive syllogism: $A, \neg A \vee B/B$, for taking an evaluation π that makes A both true and false and B false only. In fact, the disjunctive syllogism is the only classically valid inference to fail for LP in the sense that if it is added into LP then LP collapses into classical logic [12]. It might be thought that if the disjunctive syllogism fails then the system of inference is too weak to permit reasonable conclusions [13]. The logic of ***minimal*** paradox LP_m, among other things, can overcome the drawback.

Notice that to obtain a LP counter example to the disjunctive syllogism we must render the situation inconsistent by making some formula both true and false. Thus it is natural to take consistency as a default assumption. LP_m extends LP based on the intuition that normally contradictions are rare, and we assign a truth value that is both true and false (01) to a proposition only when we are forced to do so, that is, only when the proposition is a contradiction. In fact, LP_m is based on the idea of nonmonotonic logic as a kind of ***minimally inconsistent*** entailment.[6]

Definition 2 *Let S be a set of formulas. A evaluation π of S is* minimally inconsistent (mi) *iff there is no other model π' of S such that $\pi' \prec \pi$, where the partial order $\prec$ is defined over evaluations in the following way: let π and π' be two evaluations, $\pi' \prec \pi$ iff for any atomic proposition p, if $\pi'(p) = 01$ then $\pi(p) = 01$, and there is an atomic proposition q such that $\pi(q) = 01$ but $\pi'(q) \neq 01$.*

Intutively, $\pi' \prec \pi$ iff π contains more contradictions than π' does, and the *mi*-models are those in which the contradictions would be minimal. Semantically, we can define the semantic entailment of LP_m, written as $\models_{LP_m}$, as follows.

Definition 3 *Let S be a set of formulas and A a formula, $S \models_{LP_m} A$ iff A is true in all* minimally inconsistent models *of S.*

The properties of the logic LP_m can also be seen by the following example.

Example 2. Let p and q be two different atomic propositions. It is easy to see that $p, \neg p \vee q \models_{LP_m} q$, for the evaluation π that makes $\pi(p) = 01$ would not be a *mi*-model (actually there is only one *mi*-model π that makes both $\pi(p) = 1$ and $\pi(q) = 1$); but $\neg p \vee q, p, p \wedge \neg p \not\models_{LP_m} q$, for under the evaluation π such that $\pi(p) = 01$ and $\pi(q) = 0$, $p \wedge \neg p$ is true and q is not true.

It is clear that LP_m is nonmonotonic from the example. LP_m has some nice properties. It can give all classical consequences if the premises are consistent. This can be proved by noting that the *mi*-models of consistent premises are exactly classical models, that is, there are no assignments of 01 to any proposition of consistent

[6] There is much literature on the topic of nonmonotonic logic (*cf.*, [5][9]).

premises. Moreover, LP_m still validates the disjunctive syllogism even in some inconsistent situations. For example, $p, \neg p \vee q, r \wedge \neg r \models_{LP_m} q$. That is, LP_m validates all classical inferences except where inconsistency would make them naturally doubtful anyway [13]. The set of logical truths of LP_m is exactly that of classical logic. In the other words, LP_m is equivalent to classical logic when there is not direct effect of a contradiction.

Another interesting property of LP_m is that there is no greater danger of collapse into triviality with $\models_{LP_m}$ than with $\models_{LP}$ [13]. That is, for any formula S, if there is a minimally inconsistent model of S, then $S \models_{LP} A$ is true for every A iff $S \models_{LP_m} A$ is true for every A. Furthermore, LP_m has the well-founded property. That is, if π is a model of S then there is a minimal model π' with respect to the partial order $\prec$ [9].

Notice that LP_m is nonmonotonic only in a sense that the inconsistency is minimal. As pointed out in [9], LP_m is not enough to capture the reasoning with incomplete information as motivated in nonmonotonic logic. However, it is interesting to note that nonmonotonicity yields a solution to the weakness of paraconsistent logic.

3 TABLEAUX FOR *LP*

From the examples of LP we have seen that by allowing a proposition to be *both true and false*, LP destroys the triviality of classical logic: a contradiction does not imply everything. From the viewpoint of proof theory, we can get rid of the trivial problem to invalidate the proof that everything follows from a single contradiction in classical logic anyway. Perhaps the simplest way is by using the rule of *reductio ad absurdum*. According to this rule, for any formula A, A entails B iff we can infer a contradiction from $A \wedge \neg B$. Thus if A already is contradictory, then certainly for any B we can infer a contradiction from $A \wedge \neg B$; therefore A implies B. As mentioned above, *reductio ad absurdum* fails in LP. But, as presented in [3] and [7], one way to invalidate the proof of triviality is to restrict the rule of *reductio ad absurdum* as: A entails B iff we can infer a contradiction from A and $\neg B$ by *using some relevant information from B*.

The rule of *reductio ad absurdum* is just the basis of analytic tableaux [15]. We have seen in LP, the central point of paraconsistency is that if A contains a single contradiction $p \wedge \neg p$, then it could only infer either the contradiction $p \wedge p$ or the propositions p or $\neg p$ concerning the aspects of the contradiction, and not infer anything. If we would invalidate that any B followed from a single contradiction A, we could render the closed conditions of the tableaux by using the relevant information from B, *i.e.*, to check whether B is the aspects of the contradiction or not, such that B follows from A only if it is concerning with p and/or $\neg p$. With this idea of using some relevant information from B in mind, we can formulate the proof procedure by remaining the standard rules of tableaux for classical logic but modifying the closure definition of the tableaux to get rid of the triviality. One way of formalizing the rule of *restricted* reduction to absurdity is to split the tree of tableaux T for $A \wedge \neg B$ into two trees t_1 and t_2 of tableaux for A and $\neg B$ respectively, and define *mated* tableaux to arrive on (*cf.*, [7]). Another way to do so is by using *signed* tableaux as we will do in the following way.

Definition 4 *We introduce two symbols T (for true) and F (for false) in* L. *A* signed formula *has the form* $T\psi$ *or* $F\psi$, *where* $\psi \in L$. *We extend an evaluation* π *to signed formulas as follows: for every formula A,*

1. $\pi(TA) = 1$ *iff* $\pi(A) = 1$; $\pi(TA) = 01$ *iff* $\pi(A) = 01$
2. $\pi(FA) = 1$ *iff* $\pi(\neg A) = 1$; $\pi(FA) = 01$ *iff* $\pi(\neg A) = 01$.

In the other words, $\pi(TA) = true$ iff $\pi(A) = true$, and $\pi(FA) = true$ iff $\pi(\neg A) = true$. Notice that we extend *LP* evaluation to signed formulas similar to the way of classical logic [15]. In the following, we say that a *signed literal* is a signed formula in which the objective formula of symbols T or F is an atomic proposition or negation of an atomic proposition.

Definition 5 *A tableau is a tree whose nodes are formulas. Let* ψ *be a set of signed formulas. The* signed tableaux *for* ψ *are defined inductively according to the following rules:*

1. *The tableau with* ψ *as its only node is a signed tableau for* ψ.
2. *Let* T *be a signed tableau for* ψ, *and b a branch of* T.
   (a) *If* $T\neg\neg\alpha$ *is a node of b, then the tableau obtained from* T *by extending b to* $T\alpha$ *is also for* ψ; *If* $F\neg\neg\alpha$ *is a node of b, then the tableau obtained from* T *by extending b to* $F\alpha$ *is also for* ψ;

   (b) *If* $T(\alpha \vee \beta)$ *is a node of b, then the tableau obtained from* T *by adding* $T\alpha$ *and* $T\beta$ *as two children of b's leaf is also for* ψ; *If* $F\neg(\alpha \vee \beta)$ *is a node of b, then the tableau obtained from* T *by adding* $F\neg\alpha$ *and* $F\neg\beta$ *as two children of b's is also for* ψ.

   (c) *If* $T\neg(\alpha \vee \beta)$ *is a node of b, then the tableau obtained from* T *by extending b to* $T\neg\alpha$ *and* $T\neg\beta$ *is also for* ψ; *If* $F(\alpha \vee \beta)$ *is a node of b, then the tableau obtained from* T *by extending b to* $F\alpha$ *and* $F\beta$ *is also for* ψ.

   (d) *If* $T(\alpha \wedge \beta)$ *is a node of b, then the tableau obtained from* T *by extending b to* $T\alpha$ *and* $T\beta$ *is also for* ψ; *If* $F\neg(\alpha \wedge \beta)$ *is a node of b, then the tableau obtained from* T *by extending b to* $F\neg\alpha$ *and* $F\neg\beta$ *is also for* ψ.

   (e) *If* $T\neg(\alpha \wedge \beta)$ *is a node of b, then the tableau obtained from* T *by adding* $T\neg\alpha$ *and* $T\neg\beta$ *as two children of b's leaf is also for* ψ; *If* $F(\alpha \wedge \beta)$ *is a node of b, then the tableau obtained from* T *by adding* $F\alpha$ *and* $F\beta$ *as two children of b's is also for* ψ.

Similarly, other connectives can be defined in the standard way. We say that a formula is *marked* when a rule is applied to it; otherwise, a formula is said *non-marked.*

Definition 6 *A branch b of a signed tableau* T *is* complete *iff non-marked formulas in b are signed literals, that is, every rule that can be used to extend b has been applied at least once. A signed tableau* T *for ψ is* complete *iff every branch of* T *is complete.*

Definition 7 *A branch b of a signed tableau* T *is* closed *iff one of the following conditions holds:*

1. *for some formula A, $TA \in b$ and $FA \in b$;*
2. *for some formula A, $FA \in b$ and $F\neg A \in b$.*

A signed tableau T *is* closed *iff every branch of* T *is closed.*

Comparing with the standard tableaux for classical logic, if we add a condition that a branch contains TA and $T\neg A$ is closed into the avove definition, then the tableaux are *refutation* complete about classical logic in the sense that for any signed formula ψ, ψ is contradictory iff there is a tableau for ψ such that every branch of the tableaux is closed. Therefore, according to the rule of *reductio ad absurdum*, A entails B iff we can draw a tableau for TA and FB such that every branch of the signed tableau is closed.

As mentioned above, one way to obtain a paraconsistent logic is by restricting the rule of *reductio ad absurdum*. In the case of LP, we only need to weak the closed conditions as defined above. This leads to the following sound and complete theorem of signed tableaux with respect to the semantics of LP.

Theorem 1 *Let S be a set of formulas and A a formula. $S \models_{LP} A$ iff the signed tableaux for TS and FA are closed.*

The proof of this theorem will be given in the appendix.

We write $S \vdash_{LP} A$ to denote that the tableaux for TS and FA are closed. [7]

Example 1 (Continued). Let p, q be as in Example 1. It is easy to check that $p \wedge \neg p \vdash_{LP} p$, for the tableau for $T(p \wedge \neg p)$ and Fp has only a branch $\{Tp, T\neg p, Fp\}$ which is closed; ; but $p \wedge \neg p \not\vdash_{LP} q$, for the tableau for $T(p \wedge \neg p)$ and Fq has only a branch $\{Tp, T\neg p, Fq\}$ which is not closed.

4 TABLEAUX FOR LP_m

As noted in Section 2, LP_m is a kind of minimal entailment. In order to capture LP_m by tableaux, having to prove that A is minimally entailed by S we might modify the construction of the signed tableaux to eliminate the branches of *non-minimally-inconsistent models* of S. LP_m can be captured by the minimality of eliminating the *redundant* branches over *complete* signed tableaux.

Let t be a complete branch of a tableau. We write $\Pi(t) = \{Tp | p$ is an atomic proposition, $Tp \in t$ and $T\neg p \in t\}$.

[7] Perhaps the reader could describe the procedure clearly by expoiting the tree-like structure of signed tableaux.

Definition 8 *A branch b of signed tableau* T *is* redundant, *iff b is complete and there is another complete branch b' of* T *such that $\Pi(b') \subset \Pi(b)$. We say that a non-redundant branch of a signed tableau is a minimal branch of the tableau. The* minimal *tableaux T_m for ψ is taken from the signed tableaux* T *for ψ by eliminating every redundant branch of* T.

Definition 9 *A signed tableaux* T *for ψ is* minimally closed *iff every non-redundant branch of* T *is closed, that is, the minimal tableaux T_m for ψ is closed.*

Parallel to Theorem 1, we have the following sound and complete theorem of minimal tableaux with respect to the semantics of LP_m.

Theorem 2 *Let S be a set of formulas and A a formula. $S \models_{LP_m} A$ iff the signed tableaux for TS and FA is minimally closed.*

Again the proof of this theorem will be given in the appendix.

We write $S \vdash_{LP_m} A$ to denote that the minimal tableaux for TS and FA is closed.

Example 2 (Continued). Let p, q and r be as in Example 2 . It is easy to check that $p, \neg p \vee q \vdash_{LP_m} q$, since there are two branches in which $b_1 = \{Tp, T\neg p, Fq\}$ is redundant and $b_2 = \{Tp, Tq, Fq\}$ is closed; $p, \neg p \vee q, p \wedge \neg p \not\vdash_{LP_m} q$, since non-redundant one of two branches $b_1 = \{Tp, T\neg q, Fq\}$ is not closed (the other branch $b_2 = \{Tp, Tq, T\neg p, Fq\}$ is redundant because $T(b_1) \subset T(b_2)$); and $p, \neg p \vee q, r \wedge \neg r \vdash_{LP_m} q$, since there are two branches, the one $b_1 = \{Tp, T\neg p, Fq, Tr, T\neg r\}$ is redundant and the other $b_2 = \{Tp, Tq, Fq, Tr, T\neg r\}$ is closed.

5 CONCLUSION

In summary, LP, as a paraconsistent logic, destroys the triviality for formalizing reasoning in the presence of inconsistency, but it invalidates some classical inferences that seem too weak to permit reasonable conclusions. The logic of minimal paradox, LP_m, as a nonmonotonic extension of paraconsistent logic, among other things, can overcome the drawback of LP with some nice properties in a significant way. Actually, most paraconsistent logics have the properties similar to LP. Therefore the technique of LP_m can be applied to extend other paraconsistent and relevance logics and obtain the nice properties similar to LP_m. In particular, the signed tableaux for the first-degree entailment of relevance [1] based on a four-valued semantics can be obtained by slightly changing the closed conditions of the tableaux. Futhermore, the minimal tableaux developed in this paper can also be used as satisfactory proof theories for some paraconsistent and nonmonotonic logics. For example, [9] provides the semantics of logic LP_c by combining LP with the ability of circumscription for nonmonotonic reasoning, which is *truly* paraconsistent and nonmonotonic. The minimal tableaux for LP_c can be obtained by slightly modifying the definition of redundant branches.

The main results of this paper can be straightforwardly extended into first-order case. It is not difficult to obtain the signed tableaux for first-order LP and its sound and complete theorems. The minimal tableaux for first-order LP_m can also be defined. Surprisingly, we could also obtain the sound and complete theorems of the minimal tableaux for first-order LP_m. Notice that the sound and complete theorems of the minimal tableaux for circumscription is only partially correct in first-order case. This is due to the nonmonotonicity of LP_m is weaker than most nonmonotonic logics, and hence LP_m has the well-founded property. Therefore we hope to obtain better results for first-order LP_m than the case of nonmonotonic logic, and shed some new lights on the applications of LP_m in formalizing commonsense reasoning in the future.

Appendix: Proofs of Theorems

Theorem 1 Let S be a set of formulas and A a formula. $S \models_{LP} A$ iff $S \vdash_{LP} A$, i.e., the signed tableaux T for TS and FA are closed.

Proof. Suppose that T is a signed tableau for TS and FA which is closed. We prove that $S \models_{LP} A$. Let π be a model of S. We need to prove that π is also a model of A. Suppose otherwise, π is not a model of A, then $\pi(\neg A) = 1$, and $\pi(FA) = 1$. Since T is a signed tableau for TS and FA and $\pi(FA) = 1$, it is straightforward to see that there is a branch b of complete T such that π is a model of every formulas and $\pi(FA) = 1$ in b by induction on the construction of T. But this is impossible because by the definition of closure, either there might be some atomic proposition p such that $Tp \in b$ and $Fp \in b$, and hence $\pi(Tp) = 1$ and $\pi(Fp) = 1$ in b, or else there might be some atomic proposition p such that $Fp \in b$ and $F\neg p \in b$, and hence $\pi(Fp) = 1$ and $\pi(F\neg p) = 1$ in b, this is a contradiction. From the contradiction we conclude that π must be a model of A.

Conversely, suppose that $S \models_{LP} A$. We prove that there is a signed tableau T for TS and FA such that T is closed. We claim that T is closed. Suppose otherwise, then there is a branch b of complete T such that b is not closed. Construct an evaluation π as follows: for any atomic proposition p,

1. if $Tp \cup Fp \in b$, then $\pi(p) = 01$,
2. if $Tp \in b$ but $T\neg p \notin b$, then $\pi(p) = 1$,
3. if $Fp \in b$ but $F\neg p \notin b$, then $\pi(p) = 0$,
4. otherwise, assign $\pi(p)$ indifferently.

It is straightforward to see that π is a model of every formulas (including S), and for any formula ψ (including $\neg A$), $\pi(\psi) = 1$ by induction on the complexity of formulas.

Thus π is a model of S, but is not a model of A, a contradiction with the assumption $S \models_{LP} A$, and we conclude that T must be closed.

As a corollary to the proof, we have the following simple proposition.

Corollary 1 Let S be a set of formulas. $S \models_{LP} A$ for every formula A iff for any complete signed tableau T of TS, any atomic proposition p, and any branch b of T, both Tp and $T\neg p$ are in b.

Theorem 2 Let S be a set of formulas and A a formula. $S \models_{LP_m} A$ iff $S \vdash_{LP_m} A$, *i.e.*, the signed tableaux T for TS and FA is minimally closed.

Proof. Suppose there is a signed tableau T for TS and FA such that T is minimally closed. We prove that $S \models_{LP_m} A$. Let π be a minimally inconsistent model of S. We need to prove that π is a model of A. Suppose otherwise, π is not a model of A, then $\pi(\neg A) = 1$, and $\pi(FA) = 1$. Since T is a signed tableau for TS and FA and $\pi(FA) = 1$, it is straightforward to see that there is a branch b of complete T such that π is a model of every formulas in b by induction on the construction of T. We claim that b is not closed. Otherwise, as in the case of Theorem 1, this is impossible because by the definition of closure, either there might be some atomic proposition p such that $Tp \in b$ and $Fp \in b$, and hence $\pi(Tp) = 1$ and $\pi(Fp) = 1$ in b, or else there might be some atomic proposition p such that $Fp \in b$ and $F\neg p \in b$, and hence $\pi(Fp) = 1$ and $\pi(F\neg p) = 1$ in b, this is a contradiction. We claim that b is not redundant. Otherwise, for some atomic proposition $p \in b$, we have another branch b' of T such that $\Pi(b') \subset \Pi(b)$. Defining an evaluation π' as follows: for any atomic proposition p,

1. if $Tp \cup T\neg p \in b'$, then $\pi'(p) = 01$,
2. if $Tp \in b'$ but $T\neg p \notin b'$, then $\pi'(p) = 1$,
3. if $Fp \in b'$ but $F\neg p \notin b'$, then $\pi'(p) = 0$,
4. otherwise, assign $\pi'(p)$ indiffrently.

It is straightforward to see that π' is a model of S by induction on the complexity of formulas. If $\pi'(p) = 01$ then $Tp \cup T\neg p \in b'$, but since $\Pi(b') \subset \Pi(b)$ we have $Tp \cup T\neg p \in b$ and hence $\pi(p) = 01$. By $\Pi(b') \subset \Pi(b)$ there is an atomic proposition q such that $Tq \cup T\neg q \in b$ but $Tq \cup T\neg q \notin b'$. We have $\pi(q) = 01$ but $\pi'(q) \neq 01$, and $\pi' \prec \pi$, contadicting the supposed minimality of π. From the contradictions we have that every tableau T for TS and FA contains a branch b which is neither closed nor redundant, and conclude that π must be a model of A.

Conversely, suppose that $S \models_{LP_m} A$. We prove that there is a signed tableau T for TS and FA such that T is minimally closed. We claim that T is minimally closed. Suppose otherwise, then there is a branch b of complete T such that b is neither closed nor redundant. Since b is not closed, we construct an evaluation π as follows: for any atomic proposition p,

1. if $Tp \cup T\neg p \in b$, then $\pi(p) = 01$,
2. if $Tp \in b$ but $T\neg p \notin b$, then $\pi(p) = 1$,
3. if $Fp \in b$ but $F\neg p \notin b$, then $\pi(p) = 0$,
4. otherwise, assign $\pi(p)$ indifferently.

It is straightforward to see that π is a model of every formulas (including S), and for any formula ψ (including $\neg A$), $\pi(\psi) = 1$ by induction on the complexity of formulas. Thus π is a model of S, but is not a model of A. We claim that π must be minimal. For otherwise, there is a model π' of S such that $\pi' \prec \pi$. Then there is a branch b' of complete T such that π' is a model of every formulas in b'. Let $Tp \cup T\neg p \in b'$ then $\pi'(p) = 01$, and this implies $\pi(p) = 01$ for $\pi' \prec \pi$, and hence $Tp \cup T\neg p \in b$. Similarly, there is an atomic proposition q such that $\pi(q) = 01$ but $\pi'(\neg q) \neq 01$, and hence we have $Tq \cup T\neg q \in b$ but $Tq \cup T\neg q \notin b'$. Thus we have proven that $\Pi(b') \subset \Pi(b)$. contradicting the hypothesis that b is not redundant. Therefore, π is a minimally inconsistent model of S, but is not a model of A , a contradiction with the assumption of $S \models_{LP_m} A$. We conclude that T must be minimally closed.

As a result of Theorem 1 and Theorem 2, we also prove the following *reassurance theorem* announced in [13] without proof.

Theorem 3 For a set of formulas S, if there is a minimally inconsistent of S, and $S \models_{LP} A$ is true for every A iff $S \models_{LP_m} A$ is true for every A. That is, there is no more danger of collapsing into triviality with $\models_{LP_m}$ than with $\models_{LP}$.

Proof. It is easy to see that we only need to prove that if there exists a minimally inconsistent model of S, and $S \models_{LP_m} A$ is true for any formula A, then $S \models_{LP} A$ is also true for every formula A. This is also straightforward by Corollary 1 and Theorem 2.

References

[1] Anderson. R and Belnap. N, *Entailment*, Vol. 1, (Princeton, 1975)

[2] da Costa. N and Alves. E, *A Semantic Analysis of the Calculi C_n*, Notre Dame J. of Formal Logic 15 (1977),621-630

[3] Dunn. M, *Intuitive Semantics for First-Order Entailments and 'Coupled Tree'*, Philosophical Studies, 29 (1976), 149-168

[4] Dunn. M, *A Sieve for Entailments*, J. of Philosophical Logic, 9 (1980), 41-57

[5] Ginsberg. M (Ed.), *Readings in Nonmonotonic Reasoning*, (Morgan Kaufmann, 1987)

[6] Hintikka. J, *Model Minimization - An Alternative to Circumscription*, J. of Automated Reasoning, 4 (1988)

[7] Lin. F, *Reasoning in the Presence of Inconsistency*, Proceedings of AAAI-87, (Morgan Kaufmann, 1987), 139-143

[8] Lin. F, *Tableau Systems for Logic of Paradox*, draft, 1989

[9] Lin. Z, *Circumscription in a Paraconsistent Logic*, Proceedings of Canadian Artificial Intelligence Conference, Banff, Alberta, 1994

[10] Olivetti. N, *Tableaux and Sequent Calculus for Minimal Entailment*, J. of Automated Reasoning 9 (1992), 99-139

[11] Priest. G, *Logic of Paradox*, J. of Philosophical Logic, 8 (1979), 219-241

[12] Priest. G, *Consistency by Default*, Technical Report, Automated Reasoning Project, Austrialian National University, 1988

[13] Priest. G, *Reasoning about Truth*, Artificial Intelligence, 39 (1989), 231-244

[14] Priest. G *et al.* (Eds.), *Paraconsistent Logic: Essays in the Inconsistency*, (Philosophia Verlag, 1989)

[15] Smullyan. M, *First-Order Logic*, (Springer Verlag, 1968)

EFFECTS OF ATTRIBUTE REPLICATION IN INHERITANCE SYSTEMS: SOME EMPIRICAL/SIMULATION RESULTS

Aboalfazl Salimi

Aviation Computer Science Department
Embry-Riddle Aeronautical University
Daytona Beach, Florida 32114
Phone: (904)226-6681
Fax: (904)226-6678
E-mail: salimi@erau.db.erau.edu

Fernando Gomez and Ali Orooji

Department of Computer Science
University of Central Florida
Orlando, Florida 32816
Phone: (407)823-2341
Fax: (407)823-5419
E-mail: gomez@cs.ucf.edu
E-mail: orooji@cs.ucf.edu

Abstract

Snowy-KBMS, a terminological knowledge-base management system, has been designed as an integrated AI and database system to support the new and next generation applications. In order to support inheritance on a large KB efficiently, the system employs a new strategy called *controlled and adaptable attribute search (CAAS)*. This strategy replicates attributes in the knowledge-base hierarchy to improve the search process. However, attribute replication can be controlled to balance space and time efficiency. This paper presents the CAAS mechanism in detail and the simulation results of replication effects on I/O and space requirements.

1. INTRODUCTION

Snowy-KBMS, a knowledge-base management system, has been designed as (a) a system to support the new application needs which are expected to grow in their required capabilities and the size of knowledge-base, and (b) a platform for investigating various issues that arise as the result of AI and database integration [1, 2].

Snowy-KBMS is based on KL-Snowy, a terminological knowledge representation language used by Snowy. Snowy is a system that reads expository texts, acquires knowledge from them and answers questions about the knowledge that has assimilated [3, 4, 5]. KL-Snowy allows three types of relations or attributes of a given concept: necessary, necessary and sufficient, and contingent attributes. Necessary attributes are those relations that are universally quantified, e.g., "All Americans like chocolate." Necessary and sufficient conditions are those expressed by a biconditional logic, e.g., "X is a triangle IFF X is a polygon with just three sides and X's interior angles measure 180 degrees." Necessary and sufficient relations are expressed in KL-Snowy by a special slot, called CF-slot. Finally, contingent relations are those relations that are not universally quantified, e.g., "Some Americans like chocolate." All necessary, and necessary and sufficient attributes of a given concept are inherited by its subconcepts, while contingent relations are not inherited by its subconcepts.

This paper presents an efficient inheritance algorithm, called *controlled and adaptable attribute search (CAAS)*, for Snowy-KBMS. The strategy incorporated in the algorithm is to replicate attributes in the KB hierarchy to improve the search process. However, attribute replication can be controlled to achieve an acceptable balance of space and time efficiency. CAAS allows a knowledge-base manager to make the appropriate adjustments. The CAAS mechanism and simulation results of replication effects (on I/O and space requirements) are presented in this paper. Section 2 provides an overview of Snowy-KBMS. Section 3 describes the CAAS mechanism in detail. The simulation experiments, to investigate the effects of CAAS, are presented in Section 4. Finally, Section 5 provides a summary and conclusions.

2. OVERVIEW OF SNOWY-KBMS

Snowy-KBMS has been an attempt in developing a knowledge-base management system which is capable of creating and manipulating a large amount of knowledge effectively by integrating ideas from the DB and AI communities. The system is based on the Snowy knowledge representation model which is currently being used for natural language processing, representation of scientific texts, and development of expert systems. Snowy-KBMS has the following capabilities:

- The structure of an object and in general the structure of the entire knowledge-base (or schema) can change dynamically as a result of insert, delete, update, or even query processing operations. This capability is required in terminological-based systems where an object can be identified uniquely even when it is described differently.
- A large knowledge structure can be stored on the secondary storage and can be efficiently manipulated through various data structures.

The Snowy-KBMS operations fall into two different categories. Some operations are already supported by Snowy, but are based on the representation of knowledge-base in the main memory. We have added an efficient secondary memory functionality to such operations. Some operations, on the other hand, are not supported by Snowy. Examples of these operations are delete and update. These operations have been implemented in Snowy-KBMS.

3. CONTROLLED AND ADAPTABLE ATTRIBUTE SEARCH MECHANISM

A variety of alternatives are available for representing inherited attributes of a concept. These alternatives range from no-replication to full-replication strategies [6]. In no-replication strategy, one must traverse a knowledge-base hierarchy to locate a particular attribute which is not locally defined. In a full-replication strategy, attributes are replicated in all subclasses and, therefore, no KB traversal is needed. These two extremes tradeoff time for space and vice versa.

To implement inheritance in Snowy-KBMS, we have designed a new strategy called *controlled and adaptable attribute search (CAAS).* This strategy involves replication of each universally quantified conceptual relation r of a concept C in certain subconcepts of C. The degree of replication, referred to as *hierarchy traversal distance* (HTD), determines space/time efficiency.

Let us describe the CAAS strategy with an example. Consider the sample KB shown in Fig. 1, where $HTD = 2$. In this example, concept Ci is a descendant of concept Ci-1, Ri is a local attribute of concept Ci such that Ci is universally quantified in Ri (i.e., Ri is inherited by subconcepts of Ci) and Si is a local attribute of Ci such that Ci is not universally quantified in Si (i.e., Si is not inherited by subconcepts of Ci). Now, let $HTD = 2$, i.e., the degree of attribute replication is 2. Using this value, R1 of C1 is replicated in C4, C7, C10, etc. Similarly, R2 of C2 is replicated in C5, C8, C11, etc. That is, Ri of Ci is replicated in Cj (a subconcept of Ci) where $j = k * (HTD + 1)$. In other words, subconcepts along each descendant path of a parent concept C, which will store replicated values, are HTD concepts away from each other.

Using this strategy, when the value of a conceptual relation is to be accessed, the maximum number of concepts to be visited in each ancestor path will be HTD. When an attribute is referenced, the system determines the value of the attribute by traversing the knowledge-base hierarchy starting from the current concept up to a maximum number of levels (i.e., HTD) in the concept's ancestor paths. It is guaranteed that a specified attribute can be found along the paths if the attribute is to be inherited. Again, the maximum length of such search paths is called hierarchy traversal distance (HTD).

Using the example of $HTD = 2$, when a concept Ci references an attribute Rj, the system first searches Ci for Rj. If the attribute is not defined in Ci, the first immediate ancestor of Ci, which is Ci-1, will be searched for the attribute. If the attribute is not found, the second immediate ancestor of Ci, Ci-2, will be searched. If Rj is not found in Ci-2 either, the search process terminates, indicating that Rj cannot be accessed by Ci. If Rj was to be accessible by Ci, this search process would guarantee that Rj would be found in Ci, Ci-1 or Ci-2. Thus, the search stops after the HTD-th ancestor in each search path is visited.

When $HTD = 0$, system functions in a full-replication mode, and when $HTD = MaxLevel$ (MaxLevel is the height of the knowledge-base), system functions in a no-replication mode. By adjusting the value of HTD, however, one can achieve an appropriate balance between space and time, depending on different system requirements, structures and operations.

Note that in order to support such feature, we need to systematically replicate appropriate attributes in subclasses. For example, referring to Fig. 1 again, the local attribute R1 of C1 must be replicated in C4. Otherwise, C5 and C6 would not have access to this attribute.

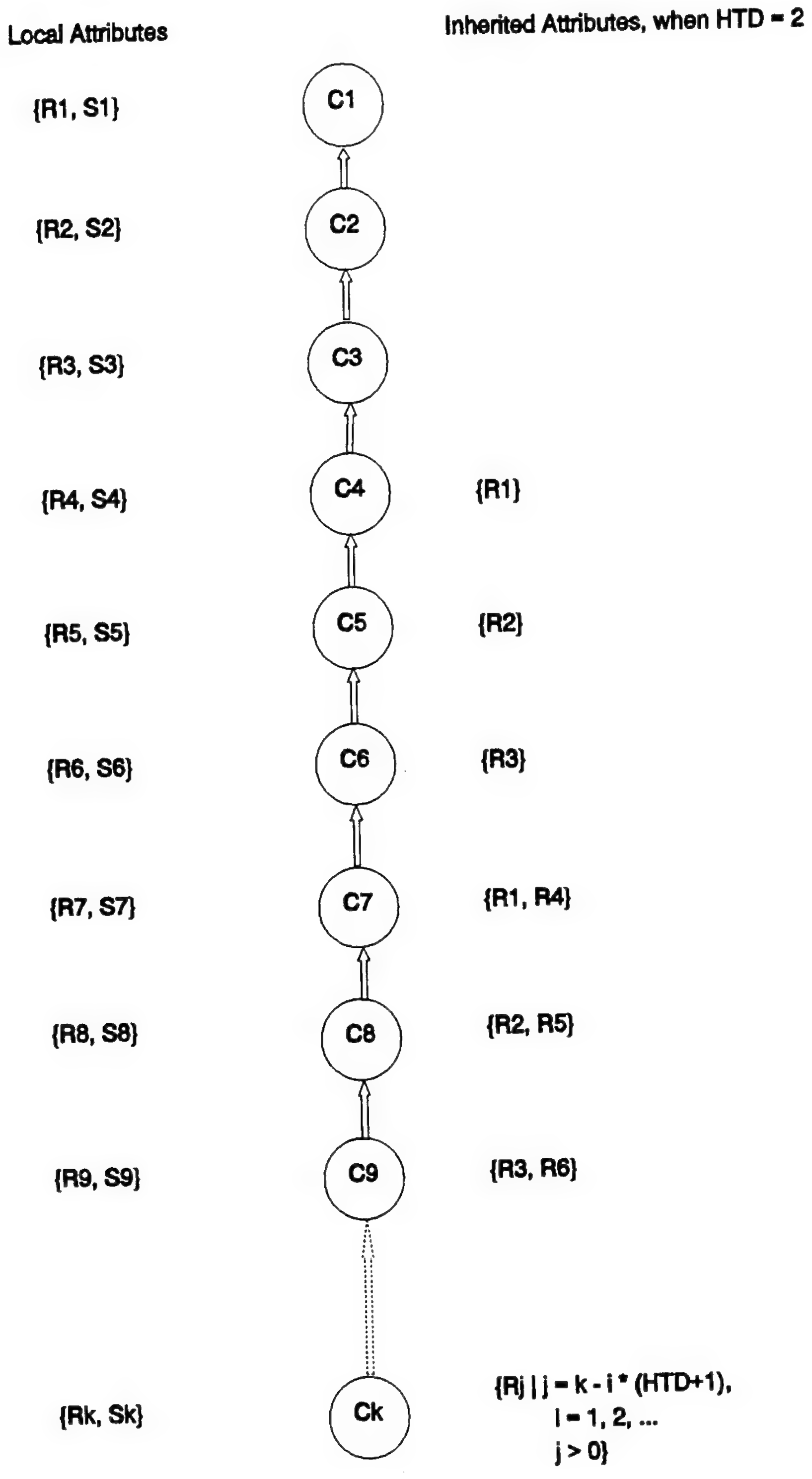


Fig. 1. Sample knowledge-base

4. SIMULATION OF HTD

In order to better understand the effectiveness of our design, we have run simulations using the Infer operation (this operation determines answers to user queries). Since other graph traversing operations (the Snowy-KBMS dominating operations) operate similar to Infer, we can conclude that our design will have comparable results for such operations. In the following sections, we will describe the simulation environment, knowledge-base structures, query mixes, simulation parameters, and overall results of the simulation.

4.1 KNOWLEDGE STRUCTURE AND QUERIES

We simulated the Infer operation on a knowledge-base of N (a simulation parameter) concepts. Each concept consists of a conceptual relation which is either universally or existentially quantified. Percentages of such conceptual relations in the KB are determined by a simulation parameter.

During each simulation run, a random DAG (directed acyclic graph) satisfying the hierarchical constraints of Snowy-KBMS is generated. Special cases such as linear and tree structures are also generated during various phases of simulation.

For each KB structure, appropriate number of queries are generated and the corresponding answers are inferred from the KB. Each query, in turn, is either universally or existentially quantified. The mix of such queries is also determined by a simulation parameter.

The set of queries is used to compare the effects of change in hierarchy traversal distance (HTD). The value of HTD changes from 0 to the maximum level of the knowledge-base. As mentioned before, a maximum and a zero value for HTD correspond, respectively, to no replication and full replication of universally quantified conceptual relations in the KB hierarchy.

4.2 SIMULATION PROCESS

The simulation process builds a knowledge-base of size N (i.e., N concepts) and for each value of HTD (from 0 to maximum knowledge-base level) processes a set of queries. After processing each set of queries, the simulation process plots the results showing the effects of HTD on I/O, HTD on space, and space on I/O.

4.3 SIMULATION PARAMETERS AND RESULTS

The goal of the simulation was to determine the effects of HTD on I/O and space requirements. Such effects must be studied in terms of other simulation parameters such as number of parents per concept, number of children and total number of attributes in the KB.

The parameters and their corresponding range of values used in the simulation process are shown in Table 1. Some of the simulation scenarios and the corresponding results are

Table 1: Simulation Parameters and Their Range of Values

Parameter	Description	Range of Values
N	Number of concepts	50, 200, 500, 1000
MinAtt–MaxAtt	Number of conceptual relations	1, 1..N/4, 1..2N/4, 1..3N/4, 1..N
%AllA	Percentage of universally quantified relations	100%, 75%, 50%, 25%, 0%
MinP–MaxP	Number of parents per concept	1, 1..3, 4..6, 7..10
MinC–MaxC	Number of children per concept	1, 1..3, 4..6, 7..10
Q	Number of queries processed in each simulation	N
%AllQ	Percentage of universally quantified Queries	100%, 75%, 50%, 25%, 0%
HTD	Hierarchy Traversal Distance	0..Max-KB-Level

summarized in the following sections. (Because of space considerations, not all scenarios and results are presented in the following sections. The reader may refer to [2] for complete details on the simulation.)

4.3.1 Effect of Number of Parents

Scenario 1 was intended to investigate the effect of change in number of parents for a concept in the KB on I/O and space requirements. In this scenario, the varying parameter was the average number of parents (fan-ins) per concept. Four ranges were used for this parameter: 1-1, 1-3, 4-6 and 7-10. The remaining parameters had the following values: N: 200, MinC-MaxC: 4-6, MaxAtt: 200, %AllA: 50, Q: 200, and %AllQ: 50. The first case (i.e., when number of parents is 1) is intended to generate a shallow KB having a tree structure.

The experiments showed that for the three KBs having non-tree structures, as the average number of parents increases, less change to HTD is required to initiate a change to I/O (Fig. 2). For example, a 70%, 50% and 35% decrease in HTD is required, respectively (for increasing number of parents), before I/O starts decreasing. However, after the change in I/O is initiated, more change in HTD value is required in order to achieve the same I/O improvement, as the number of parents increases. For example, for increasing number of parents, respectively, a 20%, 40% and 55% decrease in HTD is required in order to improve I/O by 55%. In short, as the number of parents increases, less change to HTD is required to start an improvement in I/O. However, as soon as I/O starts decreasing, more change to HTD is required in order to achieve the same I/O improvement as the number of parents increases.

For a KB having a tree structure, HTD had a relatively different behavior compared to non-tree type KBs. In this case, I/O started decreasing soon after HTD started decreasing (after about 22% decrease in HTD). Similar to other cases, however, at about 90% decrease in HTD, I/O decreased by about 55%.

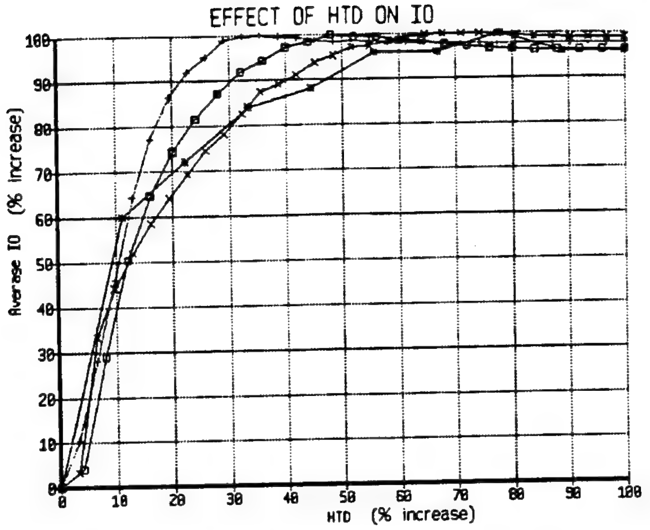


Fig. 2. I/O vs. HTD when number of parents changes

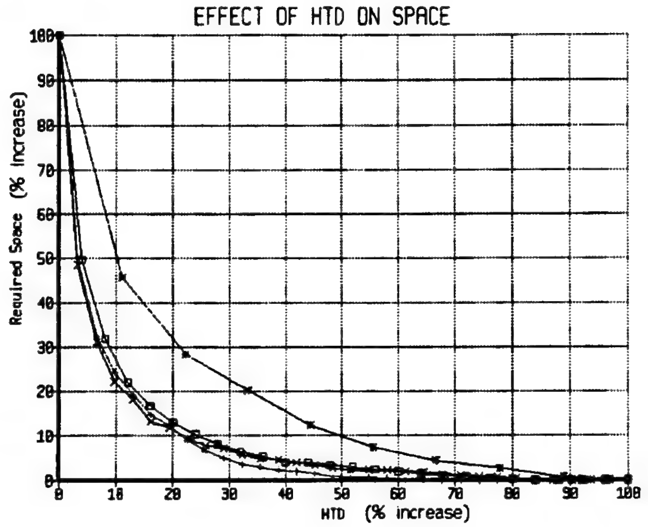


Fig. 3. Space vs. HTD when number of parents changes

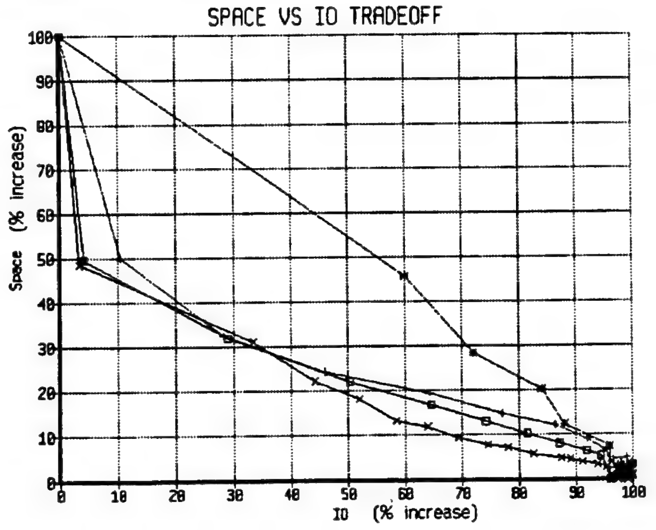


Fig. 4. Space vs. I/O when number of parents changes

An interesting finding, Fig. 3, was that even though the change in number of parents had varying effects on I/O with respect to HTD, change in HTD had the same effect on space requirement in all non-tree KBs. For example, an 80% decrease in HTD increased space requirement by only about 12%. This implies that for any non-tree KB configuration, described in this scenario, we can just focus on I/O requirements based on the number of concepts, N, and the average number of parents in the KB because space change is uniform across all configurations.

In KBs having a tree structure, decreasing HTD had immediate effect on space requirement, and space requirement increased more rapidly than non-tree cases. For example, 80% decrease in HTD increased space requirement by more than 30% in a tree-structured KB vs. 12% increase in a non-tree KB.

In all cases, decreasing HTD by more than 90% increased space requirement drastically. For example, in a non-tree KB, decreasing HTD from 90% to 100% of its maximum value (i.e., 10% decrease) increased space requirement from about 25% to 100% of its maximum value (i.e., 75% increase).

Fig. 4 shows the tradeoff between I/O and space for this scenario. All three non-tree KBs have, approximately, similar I/O and space tradeoffs (larger number of parents have offered relatively better I/O vs. space tradeoffs). In all three cases, 90%-97% improvement in I/O has been achieved by increasing space requirement by about 50% which is a good tradeoff. For a tree-structure KB, however, tradeoff is not that impressive. For example, 90% improvement in I/O has required about 90% extra space. In short, as number of parents increases, there is a better I/O vs. space tradeoff as HTD decreases. It is apparent that duplication is more effective in a KB which has a non-tree structure than a tree structure.

We conclude that for the non-tree KBs, the value of HTD needs to be about 5% of MaxLevel (the maximum level of the KB) in order to provide a reasonable tradeoff between I/O and space (e.g., 90%-97% improvement in I/O by using 50% more space). In the KB having a tree structure, there is a linear tradeoff between I/O and space when HTD value is 5%-10% of MaxLevel. That is, we can achieve a 50% improvement in I/O by sacrificing about 50% more space. In short, HTD should be in the range of 5%-10% of its maximum value for a reasonable I/O and space tradeoff.

4.3.2 Effect of Number of Children

Scenario 2 was intended to investigate the effect of change in number of children for a concept in the KB on I/O and space requirements. In this scenario, the varying parameter was the average number of children (fan-outs) per concept. Four ranges were used for this parameter: 1-1, 1-3, 4-6, and 7-10. The remaining parameters had the following values: N: 200, MinP-MaxP: 4-6, MaxAtt: 200, %AllA: 50, Q: 200, and %AllQ: 50. The first case (i.e., when number of children is 1) of this scenario is intended to generate a deep KB having a linear structure.

The results of experiments showed that, except for the case representing a linear KB, cases with larger fan-outs required more initial change to HTD to affect I/O. This result, shown in Fig. 5, is opposite of Scenario 1. This can be attributed to the fact that larger

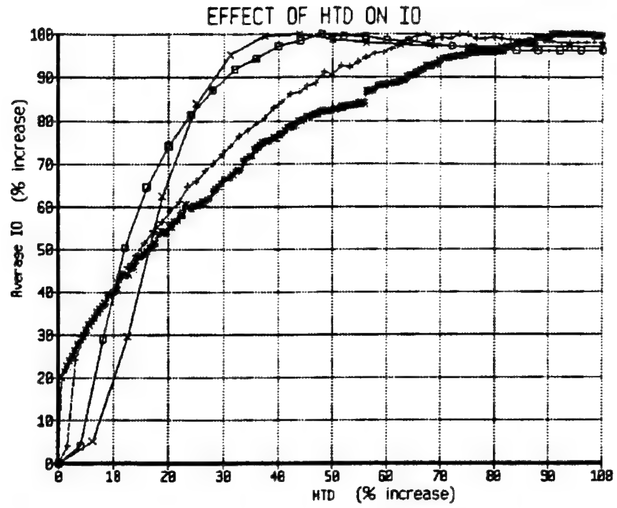


Fig. 5. I/O vs. HTD when number of children changes

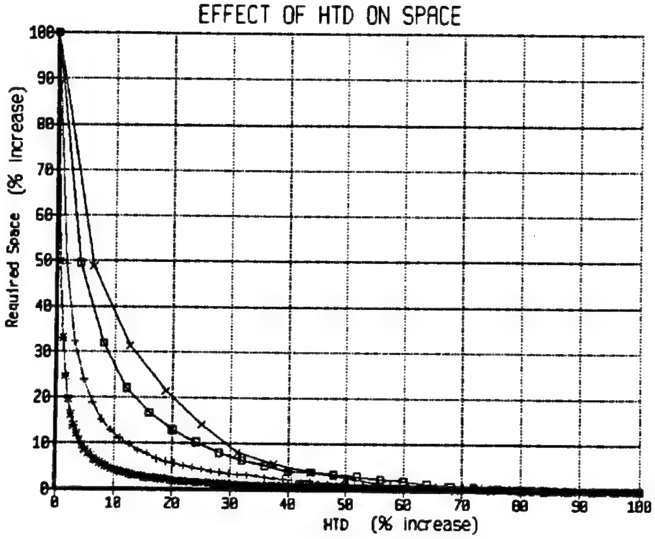


Fig. 6. Space vs. HTD when number of children changes

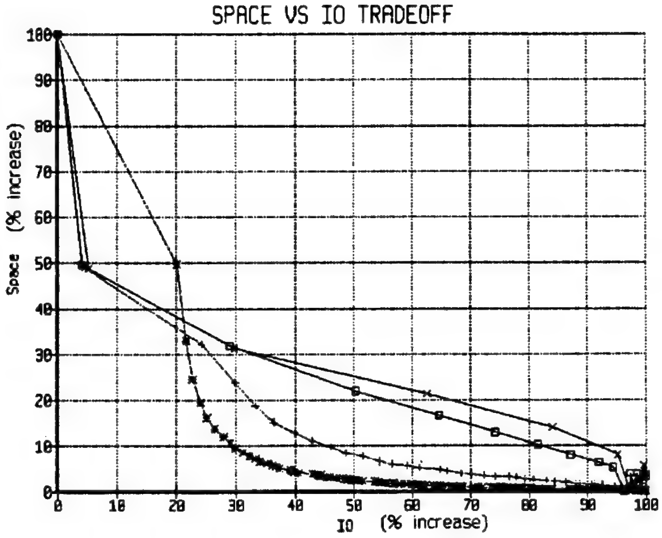


Fig. 7. Space vs. I/O when number of children changes

fan-outs generate shallower KB structures and, therefore, higher rate of change to HTD is required to start its effect on I/O. For example, a 35%, 52% and 62% decrease in HTD started its effect on I/O, for increasing values of children, respectively. On the other hand, from the point at which HTD started its effect on I/O, decreasing HTD further had greater effect on I/O as the number of children increased. This can be seen from the larger curvature of the plots in Fig. 5. For example, as the number of children increases, a 60%, 40% and 25% more decrease in HTD, respectively, decreased I/O by about 70%. In short, for larger number of children, more change to HTD is required to make an initial change to I/O, and each unit change to HTD has higher effect on I/O.

In the case of KB having a linear structure, HTD had a more uniform effect on I/O. The result was similar to that of tree-structure KB of Scenario 1. The difference, however, was that the effect of HTD on I/O in Scenario 2 was smoother than the one in Scenario 1. The reason was that in Scenario 1 the KB had a shallow but wide structure (i.e., small maximum level), but in Scenario 2 the KB had a linear structure (i.e., large maximum level). As a result, HTD had a wider range of values in Scenario 2, therefore having a smoother curve.

Fig. 6 shows that as the number of children increases, space requirement increases by a higher rate for decreasing values of HTD. For example, a 90% decrease in HTD has increased space requirement by about 5%, 12%, 27% and 40%, respectively.

As depicted in Fig. 7, cases with higher number of children offer less effective tradeoff between I/O and space requirement. For example, a 50% decrease in I/O is achieved by adding about 2%, 8%, 22% and 30% more space, respectively.

We conclude that for the non-linear KBs, when HTD has a value 2%-7% of MaxLevel, a reasonable I/O and space tradeoff is achieved (e.g., 95% I/O improvement vs. 50% extra space). For the linear KBs, a 1% of MaxLevel value for HTD provides a reasonable I/O and space tradeoff (e.g., 80% I/O improvement vs. 50% more space).

4.3.3 Effect of Number of Concepts

Scenario 3 was intended to investigate the effect of change in number of concepts in the KB on I/O and space requirements. In this scenario, the varying parameter was the number of concepts in the KB (i.e., N). We considered four different values: 50, 100, 200 and 500. The remaining parameters had the following values: MinP-MaxP: 4-6, MinC-MaxC: 4-6, MaxAtt: 200, %AllA: 50, Q: N (i.e., 50, 100, 200 and 500, respectively), and %AllQ: 50.

The results of experiments showed that as the number of concepts increases, a higher change to HTD is required to decrease I/O by the same rate. For example, Fig. 8 shows that a 45%, 47%, 52% and 62% decrease in HTD is required before I/O starts decreasing. Also, in order to achieve 50% improvement in I/O, HTD needs to be decreased by about 80%, 83%, 88% and 92%, respectively.

As depicted in Fig. 9, when the number of concepts increases, decreasing HTD does not have proportional space increase. For example, a 90% decrease in HTD has required about 50%, 40%, 28% and 15% extra space, respectively.

As the number of concepts increases, there is better I/O vs. space tradeoff for decreasing

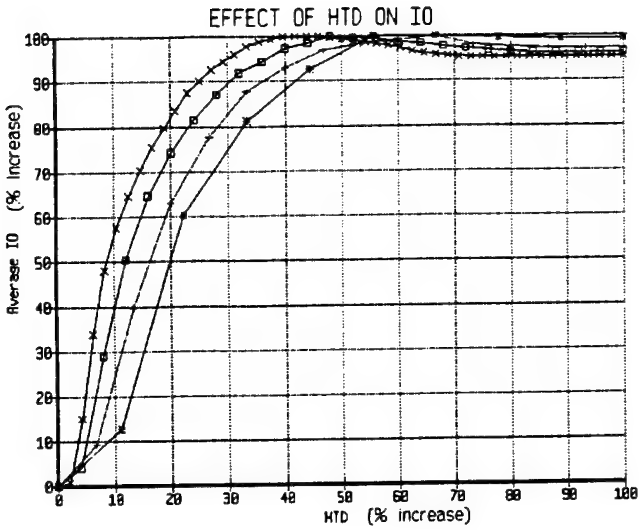


Fig. 8. I/O vs. HTD when number of concepts changes

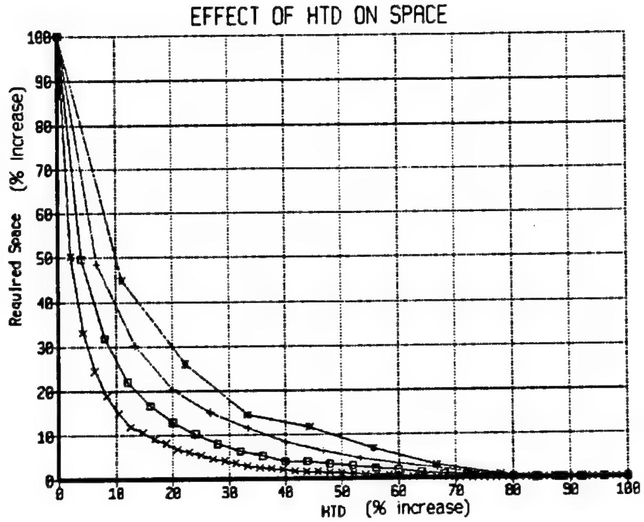


Fig. 9. Space vs. HTD when number of concepts changes

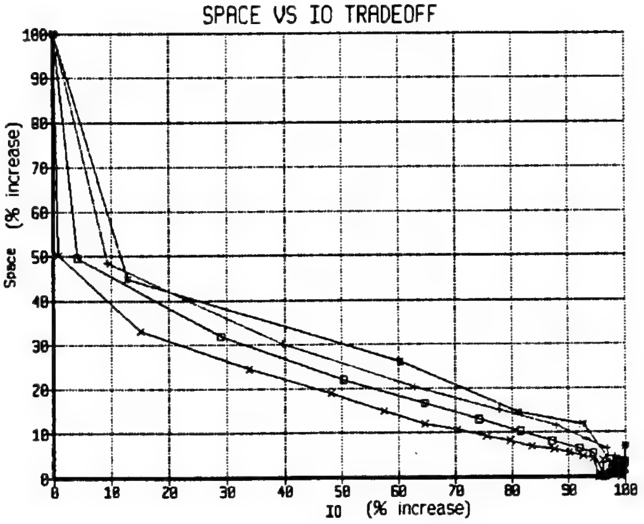


Fig. 10. Space vs. I/O when number of concepts changes

values of HTD. For example, Fig. 10 shows that I/O has been improved by about 88%, 91%, 96% and 99%, respectively, when space requirement has been increased by 50%. In summary, decreasing HTD offers better I/O vs. space tradeoff for larger number of concepts in a KB, considering the remaining parameters constant.

We conclude that an HTD value 2%-10% of MaxLevel provides a reasonable tradeoff between I/O and space (e.g., 95% improvement in I/O vs. 50% more space).

4.3.4 A General Scenario

As described before, when HTD has its maximum value, no replications of conceptual relations will take place. As HTD decreases, more space will be occupied by replicated conceptual relations. When HTD is 0, a universally quantified conceptual relation of a concept will be replicated in every descendant of the concept. In all the simulations, we processed queries for every value of HTD starting from 0 and ending with the value corresponding to the maximum KB level.

In previous scenarios, different parameters were varied to study their effects on I/O and space requirements. For each scenario, we also considered different HTD values (from no replication of conceptual relations to full replications). In Scenario 7, we kept all parameters constant at their "typical" values to concentrate specifically on HTD effects. The parameters for the general scenario had the following values: N: 500, MinP-MaxP: 4-6, MinC-MaxC: 4-6, MaxAtt: 500, %AllA: 50, Q: 500, and %AllQ: 50.

The results of experiments showed that the first 60% decrease in HTD value did not have any significant effects on I/O. As shown in Fig. 11, effects were apparent when HTD had a value in the range of 0% to 40% of its maximum value (recall that lower HTD implies higher replication).

As represented in Fig. 12, the first 60% decrease in HTD value did not have any noticeable effects on space requirement either. That is, during this decrease in HTD value, space requirement did not increase with any noticeable amount. Decreasing HTD from its 60% value to 0% value started increasing space requirement. Space requirement seems reasonable when HTD has a value in the range of 0% to about 5% of its maximum value.

There is a very interesting and reasonable tradeoff between I/O and space requirement. As shown in Fig. 13, 66%, 85% and 99% improvement in I/O has been achieved by sacrificing only 25%, 32% and 50% increase in space, respectively.

4.3.5 Concluding Remarks on the Simulation

We can draw some overall conclusions based on the simulation results. First, in order to improve I/O by any noticeable amount, HTD needs to be decreased to a value in the range of 40% to 0% of its maximum value. Second, in order to keep space requirement in the range of 0% to 50% of its maximum value, HTD can be decreased by 0% to 95% of its maximum value. Finally, 90-95% improvement in I/O can be achieved by sacrificing only 40-50% of space.

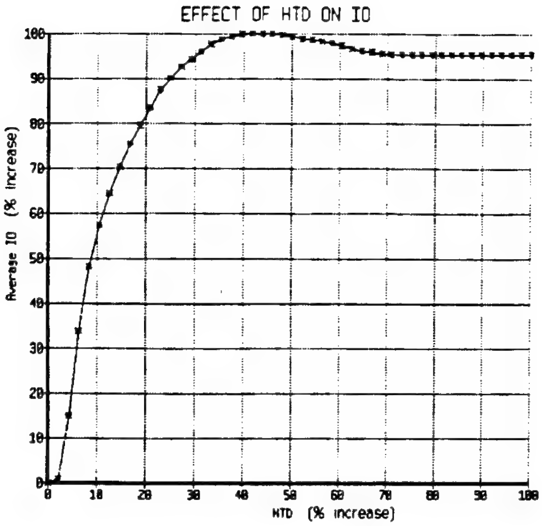


Fig. 11. I/O vs. HTD while keeping all the other parameters constant

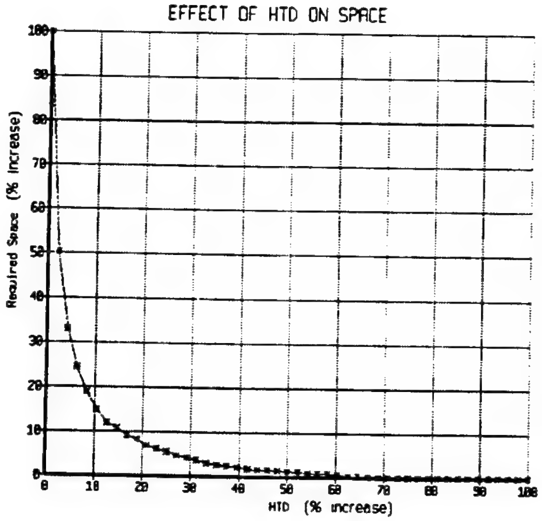


Fig. 12. Space vs. HTD while keeping all the other parameters constant

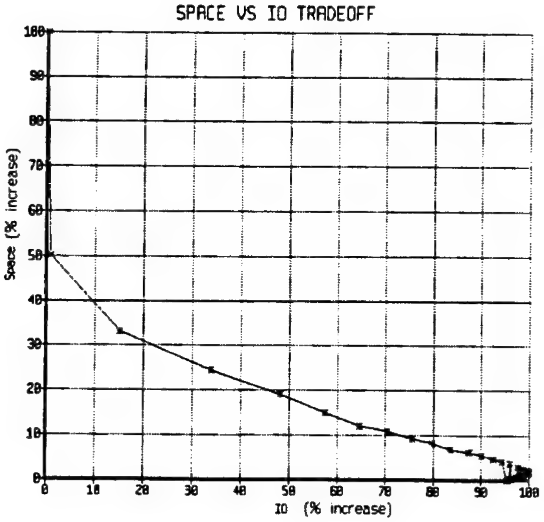


Fig. 13. Space vs. I/O while keeping all the other parameters constant

In summary, an HTD decrease of about 60% to 95% of its maximum value has shown to offer a reasonable I/O and space tradeoff. Furthermore, replication extremes (no replication and full replication) have not been effective in general.

5. CONCLUSIONS

This paper has presented an efficient inheritance algorithm, called controlled and adaptable attribute search (CAAS), for Snowy-KBMS. The algorithm replicates attributes in the KB hierarchy to improve the search process. The effects of attribute replications on I/O and space requirements have been presented using the simulation results.

Our research effort has shown how partial replication of conceptual relations can provide an acceptable I/O vs. space tradeoff in Snowy-KBMS. We feel that findings of this research effort can also be applied to other areas, such as object-oriented database and programming systems, to provide them with more efficient implementation of features such as inheritance and terminological reasoning.

References

[1] A. Salimi, F. Gomez and A. Orooji, "Extending a Scientific Text Comprehender into a Database System," Proceedings of the International Conference on Computer Application in Science, Technology and Medicine, December 1991.

[2] A. Salimi, "Snowy-KBMS: A Knowledge-Base Management System for the KL-Snowy Terminological Knowledge Representation Language," Ph.D. Dissertation (draft version), Department of Computer Science, University of Central Florida, Orlando, Florida, Summer 1994.

[3] F. Gomez and C. Segami, "The Recognition and Classification of Concepts in Understanding Scientific Texts," Journal of Experimental and Theoretical Artificial Intelligence, 1 (1989).

[4] F. Gomez and C. Segami, "Classification-Based Inferences in Retrieving Information from a Database of Scientific Facts," Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications, Miami, Florida, 1989.

[5] F. Gomez and C. Segami, "Classification-Based Reasoning," IEEE Transactions on Systems, Man, and Cybernetics, 21(3) (1991).

[6] J.G. Hughes, *Object-Oriented Databases,* Prentice-Hall, (1991).

[7] M.L. Brodie and J. Mylopoulos (Editors), *On Knowledge Base Management Systems,* Springer-Verlag, (1986).

[8] W. Kim, *Introduction to Object-Oriented Databases,* The MIT Press, Massachusetts Institute of Technology, (1990).

A LINGUISTIC GEOMETRY FOR INTELLIGENT SYSTEMS

Boris Stilman

Department of Computer Science & Engineering, University of Colorado at Denver
Campus Box 109, Denver, CO 80217-3364, USA
Email: bstilman@gothic.denver.colorado.edu

ABSTRACT

This paper reports new results on applications of Linguistic Geometry. This formal theory is intended to discover the inner properties of human expert heuristics, which were successful in a certain class of complex control systems, and apply them to different systems. The Linguistic Geometry relies on the formalization of search heuristics, which allow to decompose complex system into the hierarchy of subsystems, and thus solve intractable problems reducing the search. Currently we investigate heuristics extracted in the form of hierarchical networks of paths. The dynamic hierarchy of networks is represented as a hierarchy of formal attribute languages. This paper includes a brief survey of the Linguistic Geometry, and two examples of optimization problems for autonomous robotic vehicles. These examples demonstrate the drastic reduction of search in comparison with conventional search algorithms.

1. INTRODUCTION

It is well known that despite of the universal proliferation of computers there are many real-world problems where human expert skills in reasoning about complex systems are incomparably higher than the level of modern computing systems. At the same time there are even more areas where advances are required but human problem-solving skills can not be directly applied. For example, there are problems of planning and automatic control of autonomous agents such as space vehicles, stations and robots with cooperative and opposing interests functioning in a complex, hazardous environment. Reasoning about such complex systems should be done automatically, in a timely manner, and often in a real time. Moreover, there are no highly-skilled human experts in these fields ready to substitute for robots (on a virtual model) or transfer their knowledge to them. There is no grandmaster in robot control, although, of course, the knowledge of existing experts in this field should not be neglected – it is even more valuable. It is very important to study human expert reasoning about similar complex systems in the areas where the results are successful, in order to discover the keys to success, and then apply and adopt these keys to the new, as yet, unsolved problems. The question then is what language tools do we have for the adequate representation of human expert skills? An application of such language to the area of successful results achieved by the human expert should yield a *formal, domain independent knowledge* ready to be transferred to different areas. Neither natural nor programming languages satisfy our goal. The first are informal and ambiguous, while the second are usually detailed, lower-level tools. Actually, we have to learn how we can formally represent, generate, and investigate a *mathematical model* based on the *abstract images* extracted from the expert vision of the problem.

There have been many attempts to find the optimal (suboptimal) operation for real-world complex systems. One of the basic ideas is to decrease the dimension of the real-world system following the approach of a *human expert in a certain field*, by breaking the system into smaller subsystems. These ideas have been implemented for many problems with varying degrees of success [1, 14, 22]. Implementations based on the formal theories of linear and nonlinear planning meet hard efficiency problems [3, 11, 16, 20, 23]. An efficient planner requires an intensive use of heuristic knowledge. On the other hand, a pure heuristic implementation is unique. There is no general constructive approach to such implementations. Each new problem must be carefully studied and previous experience usually can not be applied. Basically, we can not answer the question: what are the formal properties of human heuristics which drove us to a successful hierarchy of subsystems for a given problem and how can we apply the same ideas in a different problem domain?

In the 1960's a formal syntactic approach to the investigation of properties of natural language resulted in the fast development of a theory of formal languages by Chomsky [4], Ginsburg [9], and others. This development provided an interesting opportunity for dissemination of this approach to different areas. In particular, there came an idea of analogous linguistic representation of images. This idea was successfully developed into syntactic methods of pattern recognition by Fu [7], Narasimhan [15], and Pavlidis [17], and picture description languages by Shaw [21], Feder [5], Rosenfeld [18].

Searching for the adequate mathematical tools formalizing human heuristics of dynamic hierarchy, we have transformed the idea of linguistic representation of complex real-world and artificial images into the idea of similar representation of complex hierarchical systems [24]. However, the appropriate languages should possess more sophisticated attributes than languages usually used for pattern description. The origin of such languages can be traced back to the research on programmed attribute grammars by Knuth [10],Rozenkrantz [19].

A mathematical environment (a "glue") for the formal implementation of this approach was developed following the theories of formal problem solving and planning by Nilsson [16], Fikes, Nilsson [6], Sacerdoti [20], McCarthy, Hayes [12, 13], and others based on first order predicate calculus.

To show the power of the linguistic approach it is important that the chosen model of the heuristic hierarchical system be sufficiently complex, poorly formalized, and have successful applications in different areas. Such a model was developed by Botvinnik, Stilman, and others, and successfully applied to scheduling, planning, and computer chess [1, 2].

In order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex control systems, we develop a formal theory, the so-called *Linguistic Geometry* [25-29]. This research includes the development of syntactic tools for *knowledge representation* and *reasoning* about large-scale hierarchical complex systems. It relies on the formalization of *search heuristics*, which allow one to decompose complex system into a hierarchy of subsystems, and thus solve intractable problems, reducing the search. These *hierarchical images* were extracted from the expert vision of the problem. The hierarchy of subsystems is represented as a *hierarchy of formal attribute languages.*

2. CLASS OF PROBLEMS

A ***Complex System*** is the following eight-tuple:

$$< X, P, R_p, \{ON\}, v, S_i, S_t, TR>,$$

where $X=\{x_i\}$ is a finite set of *points*; $P=\{p_i\}$ is a finite set of *elements*; P is a union of two non-intersecting subsets P_1 and P_2; $R_p(x, y)$ is a set of binary relations of *reachability* in X (x and y are from X, p from P); ON(p)=x, where ON is a partial function of *placement* from P into X; v is a function on P with positive integer values; it describes the *values* of elements. The Complex System searches the state space, which should have initial and target states; S_i and S_t are the descriptions of the *initial* and *target* states in the language of the first order predicate calculus, which matches with each relation a certain Well-Formed Formula (WFF). Thus, each state from S_i or S_t is described by a certain set of WFF of the form $\{ON(p_j)=x_k\}$; TR is a set of operators, TRANSITION(p, x, y), of transition of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed and added to the description of the state), and of WFF of applicability of the transition. Here,

Remove list: ON(p)=x, ON(q)=y;
Add list: ON(p)=y;
Applicability list: $(ON(p)=x)\wedge R_p(x, y)$,

where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out in turn with participation of elements p from P_1 and P_2 respectively; omission of a turn is permitted.

According to definition of the set P, the elements of the System are divided into two subsets P_1 and P_2. They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., $R_p(x, y)$ holds. The current location of each element is described by the equation ON(p)=x. Thus, the description of each state of the System $\{ON(p_j)=x_k\}$ is the set of descriptions of the locations of the elements. The operator TRANSITION(p, x, y) describes the change of the state of the System caused by the move of the element p from point x to point y. The element q from point y must be withdrawn (eliminated) if p and q belong to the different subsets P_1 and P_2.

The problem of the optimal operation of the System is considered as a search for the optimal sequence of transitions leading from one of the initial states of S_i to a target state S of S_t.

It is easy to show formally that robotic system can be considered as the Complex System (see below). Many different technical and human society systems (including military battlefield systems, systems of economic competition, positional games) which can be represented as twin-sets of movable units (of two or more opposing sides) and their locations, thus, can be considered as Complex Systems [25].

With such a problem statement for the search of the optimal sequence of transitions leading to the target state, we could use formal methods like those in the problem-solving system STRIPS [6], nonlinear planner NOAH [20], or in subsequent planning systems. However, the search would have to be made in a space of a huge dimension (for nontrivial examples). Thus, in practice no solution would be obtained.

We devote ourselves to the search for an approximate solution of a reformulated problem.

3. TRAJECTORY

This language is a formal description of the set of lowest-level subsystems, the set of different paths between points of the Complex System. An element might follow a path to achieve the goal "connected with the ending point" of this path.

A ***trajectory*** for an element p of P with the beginning at x of X and the end at the y of X ($x \neq y$) with a length *l* is a following string of symbols with parameters, points of X:

$$t_0=\boldsymbol{a}(x)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_l),$$

where $x_l = y$, each successive point x_{i+1} is reachable from the previous point x_i, i.e., $R_p(x_i, x_{i+1})$ holds for $i = 0, 1,\ldots, l-1$; element p stands at the point x: ON(p)=x. We denote $t_p(x, y, l)$ the set of trajectories between points x and y for the element p of the length *l*. $\mathcal{P}(t_0)=\{x, x_1, ..., x_l\}$ is the set of parameter values of the trajectory t_0.

A ***shortest trajectory*** t of $t_p(x, y, l)$ is the trajectory of minimum length for the given beginning x, end y, and element p.

Properties of the Complex System permit to define (in general form) and study formal grammars for generating the shortest trajectories. A general grammar and its application to generating the shortest trajectories for a robotic vehicle are presented in [25, 27].

A ***Language of Trajectories*** $L_t^H(S)$ for the Complex System in a state S is the set of all the trajectories of the length less than H. Different properties of this language and generating grammars were investigated in [25].

4. TRAJECTORY NETWORK

After defining the Language of Trajectories, we have new tools for the breakdown of our System into subsystems. According to the ideas presented in [1], these subsystems should be various types of trajectory networks, i.e., the sets of interconnected trajectories with one singled out trajectory called the *main trajectory*. An example of such network is shown in Fig. 1. The basic idea behind these networks is as follows. Element p_o should move along the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ to reach the ending point 5 and remove the target q_4 (an opposing element). Naturally, the opposing elements should try to disturb those motions by controlling the intermediate points of the main trajectory. They should come closer to these points (to the point 4 in Fig. 1) and remove element p_o after its arrival (at point 4). For this purpose, elements q_3 or q_2 should move along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$, respectively, and wait (if necessary) on the next to last point (7 or 9) for the arrival of element p_o at point 4. Similarly, element p_1 of the same side as p_o might try to disturb the motion of q_2 by controlling point 9 along the trajectory $\boldsymbol{a}(13)\boldsymbol{a}(9)$. It makes sense for the opposing side to include the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of element q_1 to prevent this control.

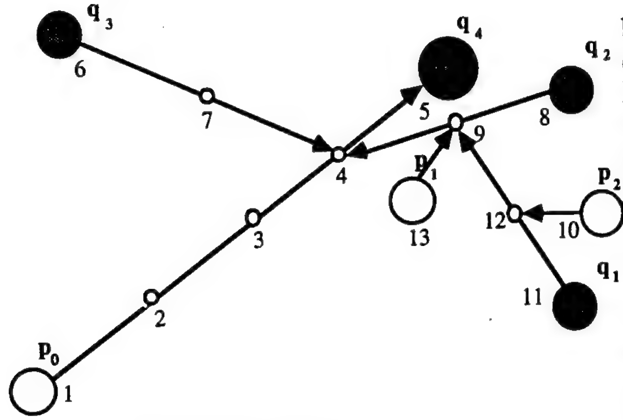


Fig. 1. A network language interpretation.

Similar networks are used for the breakdown of complex systems in different areas. Let us consider a linguistic formalization of such networks. The Language of Trajectories describes "one-dimensional" objects by joining symbols into a string employing reachability relation $R_p(x, y)$. To describe networks, i.e., "pseudo-multi-dimensional" objects made up of trajectories, we use the relation of *trajectory connection.*

For example, in Fig. 1 the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ are connected with the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ through point 4. Trajectories $\boldsymbol{a}(13)\boldsymbol{a}(9)$ and $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ are connected with $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$.

A ***Language of Zones*** $L_Z(S)$ is a trajectory network language with strings of the form

$$Z=t(p_o,t_o,\tau_o)\ t(p_1,t_1,\tau_1)\ldots t(p_k,t_k,\tau_k),$$

where $t_o,t_1,\ldots,t_k$ are the trajectories of elements $p_o,p_2,\ldots,p_k$ respectively; $\tau_o,\tau_1,\ldots,\tau_k$ are positive integer numbers (or 0) which "denote the time allotted for the motion along the trajectories" in a correspondence to the mutual goal of this Zone: to remove the target element – for one side, and to protect it – for the opposing side. Trajectory $t(p_o,t_o,\tau_o)$ is called the *main trajectory* of the Zone. The element q standing on the ending point of the main trajectory is called the *target*. The elements p_o and q belong to the opposing sides. A formal definition of this language is essentially constructive and requires showing explicitly a method for generating this language, i.e., a certain formal grammar, which is presented in [26-28].

To make it clearer let us show a formal representation of the Zone corresponding to the trajectory network in Fig. 1.

$$Z=t(p_o,\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5),\ 4)t(q_3,\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4),\ 3)t(q_2,\ \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4),\ 3)$$
$$t(p_1,\ \boldsymbol{a}(13)\boldsymbol{a}(9),\ 1)t(q_1,\ \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9),\ 2)\ t(p_2,\ \boldsymbol{a}(10)\boldsymbol{a}(12),\ 1)$$

Assume that the goal of the white side is to remove target q_4, while the goal of the black side is to protect it. According to these goals element p_o starts the motion to the target, while Black start in its turn to move their elements q_2 or q_3 to intercept element p_o. Actually, only those black trajectories are to be included into the Zone where the motion of the element makes sense, i. e., the *length of the trajectory is less than the amount of time (third parameter* τ) *allocated to it.* For example, motion along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ makes sense, because they are of length 2 and time allocated equals 3: each of the elements has 3 time intervals to reach point 4 to intercept element p_o assuming one would go along the main trajectory without move omission. According to definition of Zone the trajectories of white elements (except p_o) could only be of the length 1, e.g., $\boldsymbol{a}(13)\boldsymbol{a}(9)$ or $\boldsymbol{a}(10)\boldsymbol{a}(12)$. As far as element p_1 can intercept motion of the element q_2 at the point 9, Black include into the Zone the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of the element q_1, which has enough time for motion to prevent this interception. The total amount of time allocated to the whole bundle of black trajectories connected (directly or indirectly) with the given point of main trajectory is determined by the number of that point. For example, for the point 4 it equals 3 time intervals.

Network languages allow us to describe the "statics", i.e., the states of the System. We proceed with the description of the "dynamics" of the System, i.e., the transitions from one state to another. The transitions describe the change of the descriptions of states as the change of sets of WFF. After each transition a new hierarchy of languages should be generated. Of course, it is an inefficient procedure. To improve an efficiency of applications in a process of the search it is important to describe the change of the hierarchy of languages. A study of this change should help us in modifying the hierarchy instead of regenerating it in each state. The change may be described as a hierarchy of mappings – translations of languages. Each language should be transformed by the specific mapping called a *translation*. Translations of Languages of Trajectories and Zones are considered in [28].

5. NETWORKS FOR AUTONOMOUS AGENTS CONTROL

This class of problems can be represented as a Complex System naturally (Fig. 2). A set of X represents the operational district which could be the area of combat operation broken into smaller areas, "points", e.g., in the form of the big cube of 8 x 8 x 8, n = 512. It could be a space operation, where X represents the set of different orbits, or an air force battlefield, etc. P is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests; $R_p(x,y)$ represent moving capabilities of different robots for different problem domains: robot p can move from point x to point y if $R_p(x, y)$ holds. Some of the robots can crawl, the other can jump or ride, sail and fly, or even move from one orbit to another. Some of them move fast and can reach point y (from x) in "one step", i.e., $R_p(x, y)$ holds, others can do that in *k* steps only, and many of them can not reach this point at all. ON(p)=x, if robot p is at the point x; v(p) is the value of robot p. This value might be determined by the technical parameters of the robot. It might include the immediate value of this robot for the given combat operation; S_i is an arbitrary initial state of operation for analysis, or the starting state; S_t is the set of target states. These might be the states where robots of each side reached specified points. On the other hand S_t can specify states where opposing robots of the highest value are destroyed. The set of WFF $\{ON(p_j) = x_k\}$ corresponds to the list of robots with their coordinates in each state. TRANSITION(p, x, y) represents the move of the robot p from the location x to location y; if a robot of the opposing side stands on y, a removal occurs, i.e., robot on y is destroyed and removed.

Below we consider briefly two examples of intelligent control of robotic autonomous vehicles. They include solutions of the optimization problems for the military operation. The advantage of the Linguistic Geometry tools is in a *drastic reduction of search*. Due to the space constraints we omit all the formal details of generation of Languages of Trajectories, Zones, and Translations. These details can be found in [25-29].

5.1 AGENTS ON SURFACE

First, we consider the 2-D problem. Robots with different moving capabilities are shown in Fig. 2. The operational district X is the table 8 x 8. Squares g3, g4, d5, e6, f7 representing restricted area, e.g., neutral countries, are excluded. Robot W-FIGHTER (White Fighter) standing on h8, can move to any next square (shown by arrows). The other robot B-BOMBER from h5 can move only straight ahead, one square at a time, e.g., from h5 to h4, from h4 to h3, etc. Robot B-FIGHTER (Black Fighter) standing on a6, can move to any next square similarly to robot W-FIGHTER (shown by arrows). Robot W-BOMBER standing on c6 is analogous with the robot B-BOMBER; it can move only straight ahead but in reverse direction. Thus, robot W-FIGHTER on h8 can reach any of the points y ∈ {h7, g7, g8} in on step, i.e., while W-BOMBER can reach only c7 in one step.

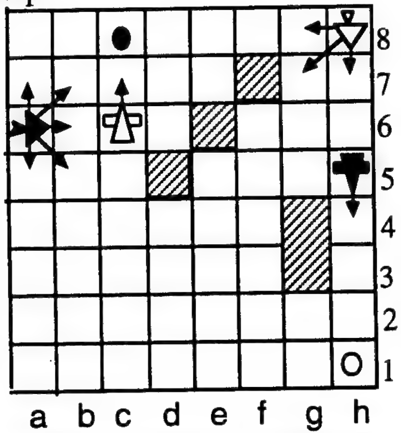


Fig. 2. The 2-D problem.

Assume that robots W-FIGHTER and W-BOMBER belong to one side, set P_1, while B-FIGHTER and B-BOMBER belong to the opposing side, P_2. Also assume that two more robots, W-TARGET and B-TARGET, (unmoving devices or targeted areas) stand on h1 and c8, respectively. W-TARGET belongs to P_1, while B-TARGET ∈ P_2. Each of the BOMBERs can destroy unmoving TARGET ahead of the course; it also has powerful weapons capable to destroy opposing FIGHTERs on the next diagonal squares ahead of the course. For example W-BOMBER from c6 can destroy opposing FIGHTERs on b7 and d7. Each of the FIGHTERs is capable to destroy an opposing BOMBER approaching its location, but it also capable to protect its friendly BOMBER approaching its prospective location. In the latter case the joint protective power of the combined weapons of the friendly BOMBER and FIGHTER can protect the BOMBER from interception. For example, W-FIGHTER located at d6 can protect W-BOMBER on c6 and c7.

The battlefield considered can be broken into two local operations. The first operation is as follows: robot B-BOMBER should reach point h1 to destroy the W-TARGET, while W-FIGHTER will try to intercept this motion. The second operation is similar: robot W-BOMBER should reach point c8 to destroy the B-TARGET, while B-FIGHTER will try to intercept this motion. After destroying the opposing TARGET the attacking side is considered as a winner of the local operation and the global battle. The only chance for the opposing side to revenge itself is to hit its TARGET on the next time interval and this way end the battle in a draw. The conditions considered above give us the description of target states of the System. The description of the initial state is obvious and follows from Fig. 2.

Assume that due to the shortage of resources (which is typical in real combat operation) or some other reasons, each side can not participate in both operations simultaneously. It means that during the current time interval, in case of White turn, either W-BOMBER or W-FIGHTER can move. Analogous condition holds for Black. Of course, it does not mean that if one side began participating in one of the operations it must complete it. Any time on its turn each side can switch from one operation to another, e.g., transferring resources (fuel, weapons, human resources, etc.), and later switch back.

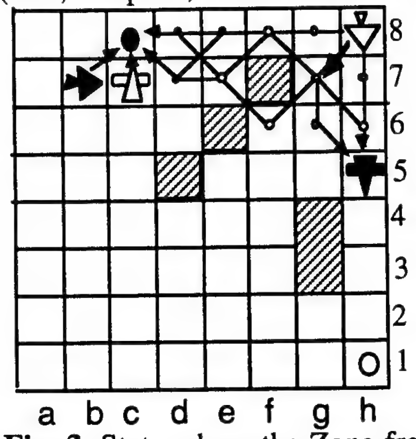


Fig. 3. State where the Zone from h8 to c8 was included into the search. Zone from h8 to h5 is shown also.

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-BOMBER from h5 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. The question is: is there a strategy for the White side to make a draw? Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. Experiments with the computer chess programs showed that for the similar problem (in chess terms - the R.Reti endgame) the search tree includes about a million moves (transitions). It is very interesting to observe the drastic reduction of search employing the Linguistic Geometry tools. In fact, the search tree generated by the grammar of searches is very close to the search tree of the R.Reti endgame generated by program PIONEER in 1977 and presented at the World Computer Chess Championship (joint event with IFIP Congress 77, Toronto, Canada). Later it was published in different journals and books, in particular, in [2].

The optimal strategy is as follows.

1. h8-g7 h5-h4 2. g7-f6. If B-BOMBER will proceed 2... h4-h3, then W-FIGHTER will turn to support the W-BOMBER: 3. f6-e7, and B-FIGHTER from a6 would never intercept W-BOMBER. For example, 3. ... h3-h2 4. c6-c7 a6-b7 5. e7-d7 or 3. ... a6-b6 4. e7-d6 h3-h2 5. c6-c7 b6-b7 6. d6-d7, and both TARGETS will be destroyed one after another.

2. ... a6-b6 3. f6-e5. Now White is ready to support W-BOMBER with the move 4. e5-d6.

3. ... b6:c6 4. e5-f4, and this way W-FIGHTER will intercept B-BOMBER on h1. The complete search tree generated by the grammar of Translations consists of 58 moves [29]. Obviously this is a drastic reduction in comparison with a million-move trees generated by conventional search procedures.

Omitting all the details we will comment on the preferred inclusion of the moves of W-FIGHTER along the diagonal: h8-g7-f6-... After evaluation of 1. c6-c7 a6-b7 2. b7:c8 the inspection procedure determined that the current minimax value (in favor of Black) can be "improved" by the improvement of the exchange on c8 which is in favor of White. This can be achieved by participation of W-FIGHTER from h8, i.e., by generation and inclusion of the new so-called "control Zone" with the main trajectory from h8 to c8 (of length 5). The set of different Zones from h8 to c8 is shown in Fig. 3. The move-ordering procedure picks the subset of Zones with main trajectories passing g7. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing W-BOMBER on h5. The motion along such trajectories allows to "gain time", i.e., to approach two goals simultaneously.

5.2 AGENTS IN SPACE

There is no specific character in the surface robotic battlefield, i.e., in the 2-D case. The Linguistic Geometry tools work analogously in the multi-dimensional cases. Consider a similar 3-D control optimization problem.

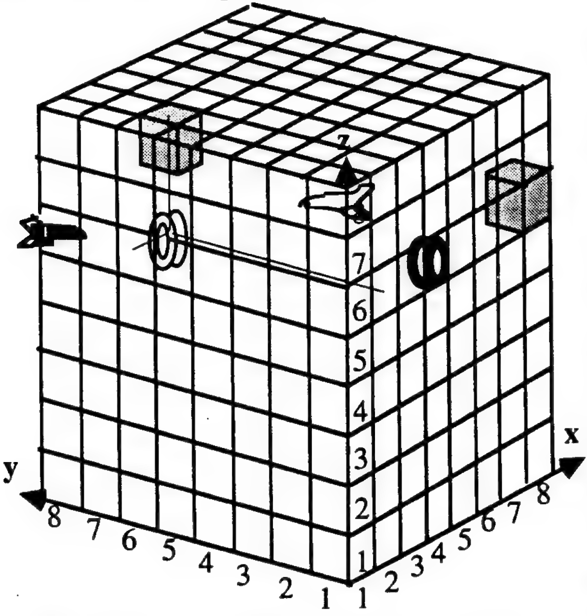


Fig. 4. A space navigation problem for autonomous robotic vehicles.

Space robotic vehicles with different moving capabilities are shown in Fig. 4. The operational district X is the cubic table of 8 x 8 x 8. Robot W-INTERCEPTOR (White Interceptor) located at 118 (x=1, y=1, z=8), can move to any next location, i.e., 117, 217, 218, 228, 227, 128, 127. The other robotic vehicle B-STATION (double-ring shape in Fig. 4) from 416 can move only straight ahead towards the goal area 816 (shaded in Fig. 2), one square at a time, e.g., from 416 to 516, from 516 to 616, etc. Robot B-INTERCEPTOR (Black Interceptor) located at 186, can move to any next square similarly to robot W-INTERCEPTOR. Robotic vehicle W-STATION located at 266 is analogous with the robotic B-STATION; it can move only straight ahead to the goal area 268 (shaded in Fig. 4). Thus, robot W-INTERCEPTOR on 118 can reach any of the points y ∈ {117, 217, 218, 228, 227, 128, 127} in one step, i.e., $R_{W\text{-}INTERCEPTOR}(118, y)$ holds, while W-STATION can reach only 267 in one step.

Assume that robots W-INTERCEPTOR and W-STATION belong to one side, while B-INTERCEPTOR and B-STATION belong to the opposing side: W-INTERCEPTOR $\in P_1$, W-STATION $\in P_1$, B-INTERCEPTOR $\in P_2$, B-STATION $\in P_2$. Also assume that both goal areas, 816 and 268, are the safe areas for B-STATION and W-STATION, respectively, if station reached the area and stayed there for more than one time interval. Each of the STATIONs has powerful weapons capable to destroy opposing INTERCEPTORs at the next diagonal locations ahead of the course. For example W-STATION from 266 can destroy opposing INTERCEPTORs at 157, 257, 357, 367, 377, 277, 177, 167. Each of the INTERCEPTORs is capable to destroy an opposing STATION approaching its location from any direction, but it also capable to protect its friendly STATION approaching its prospective location. In the latter case the joint protective power of the combined weapons of the friendly STATION and INTERCEPTOR (from any next to the STATION area) can protect the STATION from interception. For example, W-

INTERCEPTOR located at 156 can protect W-STATION on 266 and 267.

The battlefield considered can be broken into two local operations. The first operation is as follows: robot B-STATION should reach the strategic point 816 safely and stay there for at list one time interval, while W-INTERCEPTOR will try to intercept this motion. The second operation is similar: robot W-STATION should reach point 268, while B-INTERCEPTOR will try to intercept this motion. After reaching the designated strategic area the (attacking) side is considered as a winner of the local operation and the global battle. The only chance for the opposing side to revenge itself is to reach its own strategic area within the next time interval and this way end the battle in a draw. The conditions considered above give us S_t, the description of target states of the Complex System. The description of the initial state S_i is obvious and follows from Fig. 4.

Analogously with the 2-D problem, assume also that due to the shortage of resources (which is typical in real combat operation) or some other reasons, each side can not participate in both operations simultaneously. It means that during the current time interval, in case of White turn, either W-STATION or W-INTERCEPTOR can move. Analogous condition holds for Black. Of course, it does not mean that if one side began participating in one of the operations it must complete it. Any time on its turn each side can switch from one operation to another, e.g., transferring resources (fuel, weapons, human resources, etc.), and later switch back.

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-STATION from 418 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. Is there a strategy for the White side to make a draw? Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. For the similar 2-D problem the search tree includes about a million moves (transitions). Of course, in the 3-D case the search would require billions of moves. It is very interesting to observe the drastic reduction of search employing the Linguistic Geometry tools.

The details of generation of the Languages of Trajectories and Zones for the robotic systems are considered in [27].

5.3 SEARCH GENERATION FOR SPACE AGENTS

Consider how the hierarchy of languages works for the optimal control of the space robotic system introduced above (Fig. 4). We generate the search of the Language of Translations representing it as a conventional search tree (Fig. 9) and comment on its generation. In fact, this tree is close to the search tree of the relative 2-D problem [29].

First, the Language of Zones in the start state is generated. The targets for attack are determined within the limited number of steps which is called a horizon. In general, the value of the horizon is unknown. As a rule, this value can be determined from the experience of solving specific classes of problems employing Linguistic Geometry tools. In absence of such experience, first we have to consider the value of 1 as a horizon, and solve the problem within this value. If we still have resources available, i.e., computer time, memory, etc., we can increase the horizon by one. After each increase we have to regenerate the entire model. This increase means a new level of “vigilance” of the model, and, consequently, new greater need for resources.

In our case it is easy to show that within the horizons of 1, 2, 3, 4 all the models are “blind” and corresponding searches do not give a “reasonable” solution. But, again, after application of each of the consecutive values of the horizon we will have a *solution* which can be considered as an approximate solution within the available resources. Thus, let the horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of main trajectories of all Zones must not exceed 5 steps. All Zones generated in the start state are shown in Fig. 5.

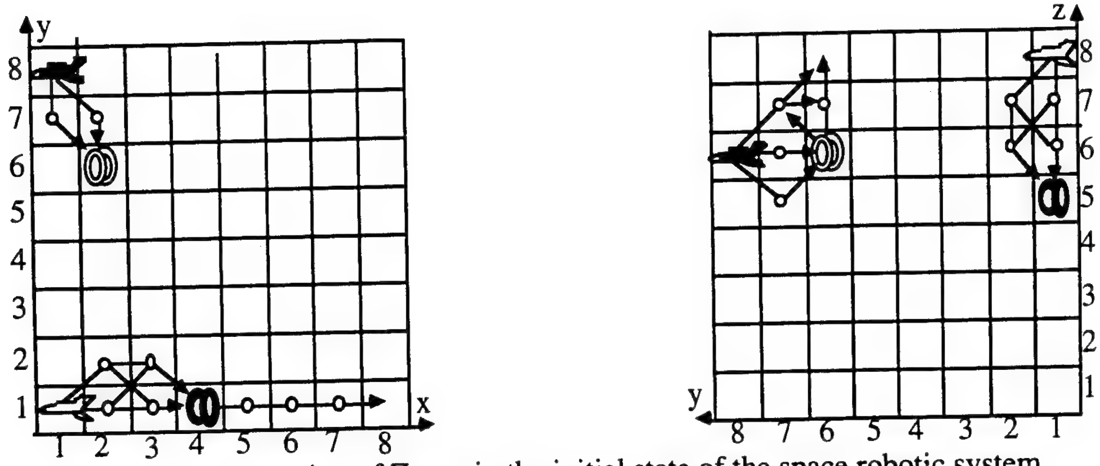


Fig. 5. Interpretation of Zones in the initial state of the space robotic system (2 projections)

These are the Zones for INTERCEPTORSs and STATIONs as attacking elements. For example, one of the Zones for W-STATION, Z_{WS}, is as follows:

Z_{WS}=*t*(W-STATION,***a***(266)***a***(267)***a***(268),3)*t*(B-INTERCEPTOR,***a***(186)***a***(277)***a***(268),3) *t*(B-INTERCEPTOR,***a***(186)***a***(276)***a***(267), 2)*t*(W-STATION, ***a***(266)***a***(277), 1)

The other trajectories of B-INTERCEPTOR, e.g., the second trajectory, ***a***(186)***a***(177)***a***(268), leading to the point 268 is included into different Zone; for each Zone only one trajectory from each bundle of trajectories with the same beginning and end is taken.

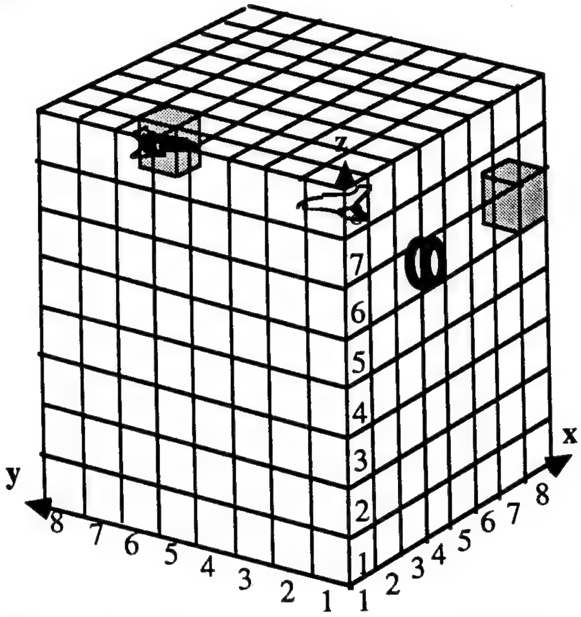


Fig. 6. The terminal state where control Zone from 118 to 268 was detected.

Generation begins with the move 1. 266-267 in the "white" Zone with the target of the highest value and the shortest main trajectory. The order of consideration of Zones and particular trajectories is determined by the grammar of translations. The computation of move-ordering constraints is the most sophisticated procedure in this grammar. It takes into account different parameters of Zones, trajectories, and the so-called chains of trajectories.

Next move, 1. ... 186-277, is in the same Zone along the first negation trajectory. The interception continues: 2. 267-268 277:268 (Fig. 6). Symbol ":" means the removal of element. Here the grammar terminates this branch with the value of -1 (as a win of the Black side). This value is given by the special procedure of "generalized square rules" built into the grammar. Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search.

After the climb up to the move 1. ... 186-277, the tree to be analyzed consists of one branch (of two plies): 2. 267-268 277-268. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange in the area 268 (in favor of the White side). This can be achieved by participation of W-INTERCEPTOR from 118, i.e., by generation and inclusion of the new so-called "control" Zone with the main trajectory from 118 to 268. The set of different Zones from 118 to 268 (the bundle of Zones) is shown in Fig. 7. The move-ordering procedure picks the subset of Zones with main trajectories passing 227. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing B-STATION on 516. The motion along such trajectories allows to "gain time", i.e., to approach two goals simultaneously.

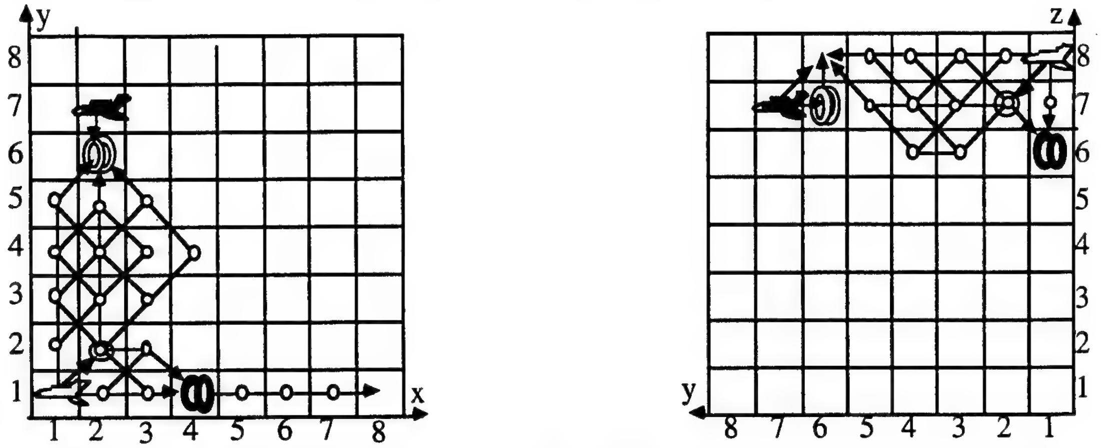


Fig. 7. Interpretation of Zones in the state where control Zone from 118 to 268 was included first (2 projections).

The generation continues: 2. 118-227 277-267. Again, the procedure of "square rules" cuts the branch, evaluates it as a win of the black side, and the grammar initiates the climb. Move 2. 118-227 is changed for 2. 118-228. Analogously with the previous case, the inspection procedure determined that the current minimax value (-1) can be improved by the improvement of the exchange on 267. Again, this can be achieved by the inclusion of Zone from 118 to 267. Of course, the best "time-gaining" move in this Zone is 2. 118-227, but it was already included (as move in the Zone from 118 to 268). The other untested move in the Zone from 118 to 267 is 2. 118-228. Obviously the grammar does not have knowledge that trajectories to 267 and 268 are "almost" the same.

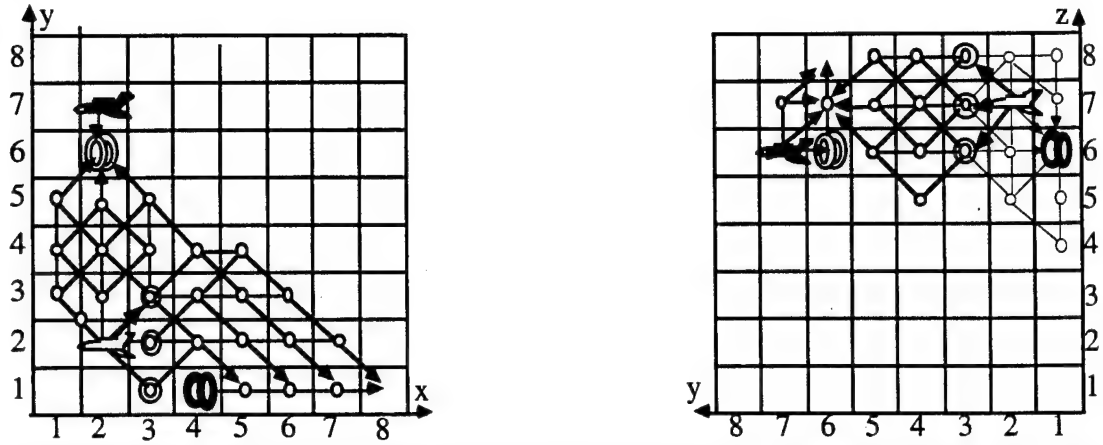


Fig. 8. Interpretation of Zones in the state where the control Zone from 227 to 267 was included first (2 projections).

After the next cut and climb, the inspection procedure does not find new Zones to

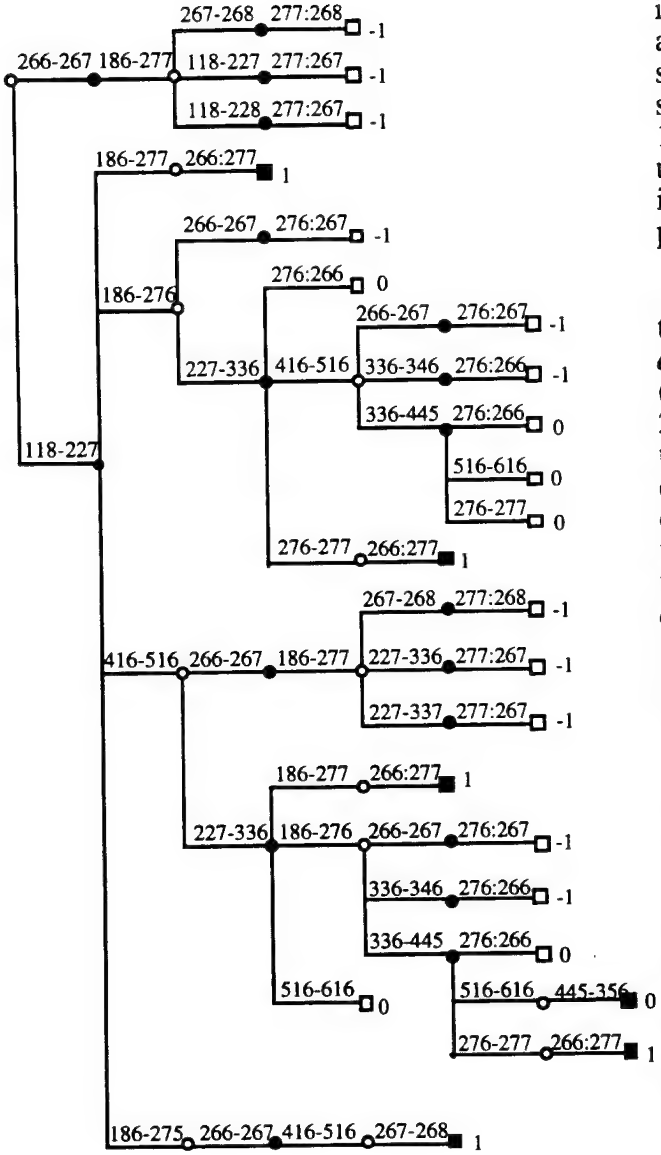


Fig. 9. Search tree for the space navigation problem.

improve the current minimax value, and the climb continues up to the start state. The analysis of the subtree shows that inclusion of Zone from 118 to 268 in the start state can be useful: the minimax value can be improved. Similarly, the most promising "time-gaining" move is 1. 118-227. The Black side responded 1. ... 186-277 along the first negation trajectories ***a***(186)***a***(277)***a***(267) and ***a***(186)***a***(277)***a***(268) shown in Fig. 5 (better see yz-projection). Obviously, 2. 266:277, and the branch is terminated. The grammar initiates the climb and move 1. ... 186-277 is changed for 1. ... 186-276 along trajectory ***a***(186)***a***(276)***a***(266). Note, that grammar "knows" that trajectory ***a***(186)***a***(276)***a***(266) is active in this state, i.e., B-INTERCEPTOR has enough time for interception. The following moves are in the same Zone of W-STATION: 2. 266-267 276:267. The "square rule procedure" cuts this branch and evaluates it as a win of the Black side.

New climb up to the move 2. ... 186-276 and execution of the inspection procedure result in the inclusion of the new control Zone from 227 to 267 in order to improve the exchange in the area 267. The set of Zones with different main trajectories from 227 to 267 is shown in Fig. 8. Besides that, the trajectories from 227 to 516, 616, 716, and 816 are shown in the same Fig. 8. These are "potential" first negation trajectories. It means that beginning with the second symbol ***a***(336), ***a***(337), ***a***(338), or ***a***(326), ***a***(327), ***a***(328), or ***a***(316), ***a***(317), ***a***(318), these trajectories become first negation trajectories in the Zone of B-STATION h5. Speaking informally, from the areas listed above W-INTERCEPTOR can intercept B-STATION (in case of white move). The main trajectories of control Zones passing one of three points, 336, 337, or 338, partly coincide with the potential first negation trajectories. The motion along such trajectories allows to "gain the time", i.e., to approach two goals simultaneously. The move-ordering procedure picks the subset of Zones with the main trajectories passing 336. Thus, 2. 227-336.

This way proceeding with the search we will generate the tree that consists of 56 moves. Obviously, this is a drastic reduction in comparison with a billion-move trees generated by conventional search procedures.

6. CONCLUSION

The example considered in this paper demonstrates the power of the Linguistic Geometry tools that allowed to transfer heuristics discovered in one problem domain, specifically, in the game of chess, to another domain of the surface and space robotic navigation. It is even more interesting that search reduction achieved in the original domain multiplied tremendously in the new domain, especially, in the 3-D case. Looking at the complexity of the hierarchy of languages which represents each state in the process of the search it is very likely that the growth from 2-D case to 3-D is linear with the factor close to one. This means that the complexity of the entire algorithm is about linear with respect to the length of the input.

The following research will lead to the development of *efficient applications* to autonomous navigation in hazardous environment, robot control, combat operations planning as well as applications in different nonmilitary engineering areas.

REFERENCES

1. Botvinnik, M.M., *Computers in Chess: Solving Inexact Search Problems.* Springer Series in Symbolic Computation, Springer-Verlag, New York, (1984).
2. Botvinnik, M., Petriyev, E., Reznitskiy, A., et al., “Application of New Method for Solving Search Problems For Power Equipment Maintenance Scheduling”, *Economics and Mathematical Methods*, **6**, 1030-1041, (1983) (in Russian).
3. Chapman, D., “Planning for conjunctive goals”, *Artificial Intelligence*, **32**(3), (1987).
4. Chomsky, N., “Formal Properties of Grammars”, *Handbook of Mathematical Psychology*, (ed. R.Luce, R.Bush, E. Galanter), Vol.2, J. Wiley, New York, 323-418, (1963).
5. Feder, J., “Plex languages”, *Information Sciences*, **3**, 225–241, (1971).
6. Fikes, R.E. & Nilsson, N.J., “STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving”, *Artificial Intelligence*, **2**, 189–208, (1971).
7. Fu, K.S. *Syntactic Pattern Recognition and Applications*, Prentice Hall, Englewood Cliffs, (1982).
8. Garey, M.R. & D.S.Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., San Francisco, (1991).
9. Ginsburg, S., *The Math. Theory of Context-Free Languages*, McGraw Hill, New York, (1966).
10. Knuth, D.E., “Semantics of Context-Free Languages”, *Mathem. Systems Theory*, **2**, 127–146, (1968).
11. McAllester, D. & Rosenblitt, D., “Systematic Non-Linear Planning”, *Proc. of AAAI-91*, 1991, pp. 634-639.
12. McCarthy, J., “Circumscription – A Form of Non-Monotonic Reasoning”, *Artificial Intelligence* , **13**, 27-39, (1980).
13. McCarthy, J. and Hayes, P.J., “Some Philosophical Problems from the Standpoint of Artificial Intelligence”, *Machine Intelligence* , **4**, 463–502, (1969).
14. Mesarovich, M.D., Macko, D., and Takahara Y., *Theory of Hierarchical Multilevel Systems.*, Academic Press, New York, (1970).
15. Narasimhan, R.N., “Syntax–Directed Interpretation of Classes of Pictures”, *Communications of the ACM* , **9**, 166–173, (1966).
16. Nilsson, N.J., *Principles of Artificial Intelligence*, Tioga Publ., Palo Alto, CA, (1980).
17. Pavlidis, T., *Structural Pattern Recognition*, Springer-Verlag, New York, (1977).
18. Rosenfeld, A. *Picture Languages, Formal Models for Picture Recognition*, Academic Press, (1979).

19. Rozenkrantz, D.J., “Programmed Grammars and Classes of Formal Languages”, *J. of the ACM*, **1**,107–131, (1969).
20. Sacerdoti, E.D., “The Nonlinear Nature of Plans”, *Proc. Int. Joint Conference on Artificial Intel.*, 1975.
21. Shaw, A.C., “A Formal Picture Description Scheme as a Basis for Picture Processing System”, *Information and Control*,**19**, 9-52, (1969).
22. Simon, H.A. *The Sciences of the Artificial*, 2-nd ed., The MIT Press, Cambridge, (1980).
23. Stefik, M., “Planning and meta-planning (MOLGEN: Part 2)”,*Artificial Intelligence*, **2**,141-169, (1981).
24. Stilman, B., “Hierarchy of Formal Grammars for Solving Search Problems”, in Artificial Intelligence. Results and Prospects, *Proceedings of the International Workshop*, Moscow, pp. 63–72, 1985, (in Russian).
25. Stilman, B. “A Linguistic Approach to Geometric Reasoning”, *Int. J. Computers and Mathematics with Applications*, **26**(7), 29-57, (1993).
26. Stilman, B. “Network Languages for Complex Systems”, *Int. J. Computers and Mathematics with Applications*, **26**(8), 51-79, (1993).
27. Stilman, B. “Syntactic Hierarchy for Robotic Systems”, *Integrated Computer-Aided Engineering*, **1**(1), 57-82, (1993).
28. Stilman, B. “Translations of Network Languages”, *Int. J. Computers and Mathematics with Applications*, **27**(2), 65-98, (1994).
29. Stilman, B. “A Formal Model for Heuristic Search”, *Proceedings of the 22nd Annual ACM Computer Science Conf.*, Phoenix, AZ, March, 1994, pp. 380-389.

A KNOWLEDGE-BASED SCENE ANALYSIS SYSTEM FOR AERIAL IMAGE

Crystal J. Su[1] Zhongquan Wu[2]

Department of Electrical Engineering
Tsinghua University
Beijing, P.R. China

1 ABSTRACT

This paper introduces a scene analysis system. The representation and maintenance of domain specific knowledge are important topics in constructing a scene analysis system. Two kinds of knowledge – about the location relations among objects and about their shapes are organized to store in a knowledge base. Location relations are represented using a semantic network and shape information is recorded in frame structures.

A target object is identified based on its location relations with reference objects and its shape information. Three sophisticated maintenance techniques are used for the system's knowledge base.

Experiments for searching some objects in an aerial image are conducted and the results are given. The maintenance of the knowledge based significantly improved the performance of the system.

2 INTRODUCTION

Image understanding plays an important role in both the investigation of the vision mechanism and many application areas, such as robotics, remote sensing, medical image interpretation, etc. Binford in [1] introduces many image analysis systems with different features.

Nagao and Marsugama [2] built a system to analyze the color of an infrared aerial image. Specific objects are recognized using various special subsystems. Which subsystem to use is based on regional features extracted from the input image. There is a "focus of attention" mechanism in this system, and it is used to select the image regions to be considered.

Selfridge [3] proposed to use the "reasoning about the success and failure" in the aerial image understanding. There are three important aspects in constructing image understanding systems: (1) select the right model for a particular image; (2) use the

[1] Current address: IBM Canada Lab, 1150 Eglinton Ave. East, North York, Ontario, Canada, M3C 1H7

[2] Current address: ESTEK Corporation, 9625 Southern Pine Blvd., Charlotte, NC, U.S.A., 28213

appropriate segmentation technique; (3) choose the proper parameters in low level processing. The paper proposed to use the reasoning about the success and failure as the general approach and developed a system to interpret scenes of suburban housing developments.

Nazif and Levine [4] solved segmentation problems using an expert system approach in natural scene understanding. A set of knowledge rules is used to segment the image into uniform regions and connected lines in their system. A set of focus rules are used to determine the order in which different areas of the image are analyzed.

The SPAM image understanding system [5] uses a rule-based approach for interpreting an airport scene. The system is composed of three components: an image/map database representing the world model, image processing tools and a rule-based system. This system can extract and identify runways, taxi routes, grassy areas and buildings from some airport images.

Human being have a lot of searching strategies that can be used in a vision system. For example, when one looks for a tiny object, it is very likely that he or she will look for big objects around the target first.

In this paper, a scene analysis system for aerial images is presented. Figure 1 is the diagram of the system. The strategy of this system is mixture of bottom-up and top-down. Data flow is passed from the low level to the high level to extract the features of objects to form object description, i.e., bottom up; Searching procedure is knowledge-directed, i.e., top-down. knowledge of the location relations among objects are used to direct the search algorithms. The following sections discusses several aspects of this system. The maintenance of the knowledge base is very important for the performance of this system, and will be discussed afterwards.

3 REPRESENTATION OF DOMAIN SPECIFIC KNOWLEDGE

Knowledge used in the system can be categorized as follows:

- Unrelated to specific image
    - common sense, e.g., a bridge is over a river.
    - specific scene knowledge, e.g., there is a bridge near a certain bridge.
- image specific knowledge, e.g., shape information and location of objects.

Now we introduce some ways to represent the following kinds of knowledge:

1. Object description;
2. Geometrical and topological relations between objects;
3. Information used to guide searching.

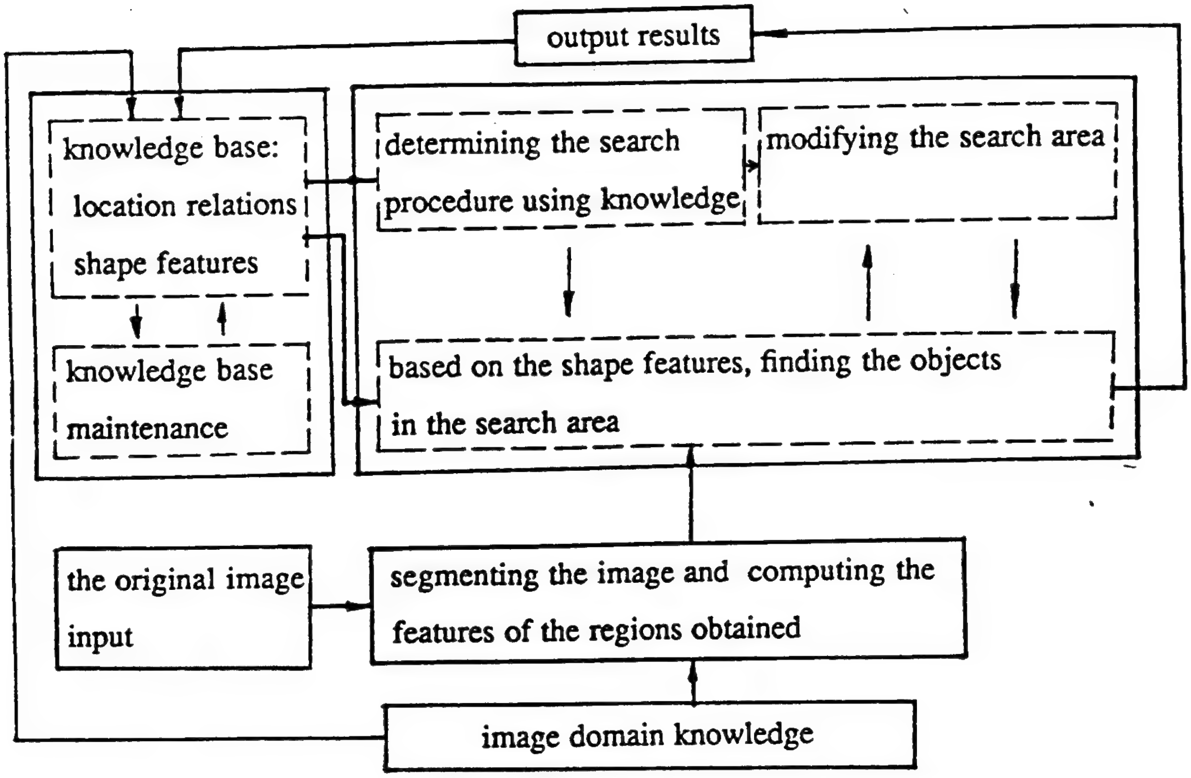


Figure 1: Block diagram of an image understanding system.

The location relations can be logically represented by a location network in terms of a directed graph, which is built on the basis of a semantic network. Let $G = (N, E)$ be a connected (or unconnected) directed graph where $N = \{n_i | i = 1, .., n\}$ is a set of nodes and $E = \{e_{ij} | i = 1, .., n; j = 1, .., n\}$ is a set of edges. The objects in the location network can then be represented by the nodes n_i in N, while the directed edge e_{ij} in E connecting a pair of nodes n_i, n_j in N is used to indicate that n_j is a reference of n_i. The semantics associated with the edge stands for the location relation of the two objects. Based on the way these reference objects affecting on the searching area, we separate these objects into several groups, and a group indicator – a small arc – is used to show the grouping. is plotted across the edges in the same group. Figure 2 shows the relations of "A is left of B, A is above C or below D". The semantics of the group indicator is "or". Suppose that the edges in the same group are painted with the same color, and the edges with no or-indicator are "colorless" edges. Note that there is an important difference between them. All the location relations represented by the "colorless" edges in the location network must be consistent with the actual configuration of the objects, whereas at least one among the location relations with the same color (i.e. in the same group) is required to be consistent with the actual configuration of the objects. An example of a location network is shown in Figure 3.

Frames are used to represent information of objects expected to be in the scene. A frame is a nested link list. There are many slots at the top level in each object frame. In this level, the slot names are the key words, see Table 1. These slots are

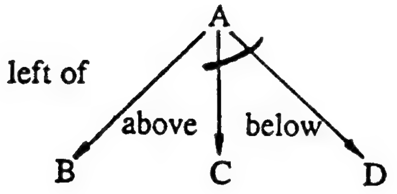


Figure 2: A location network with a group indicator.

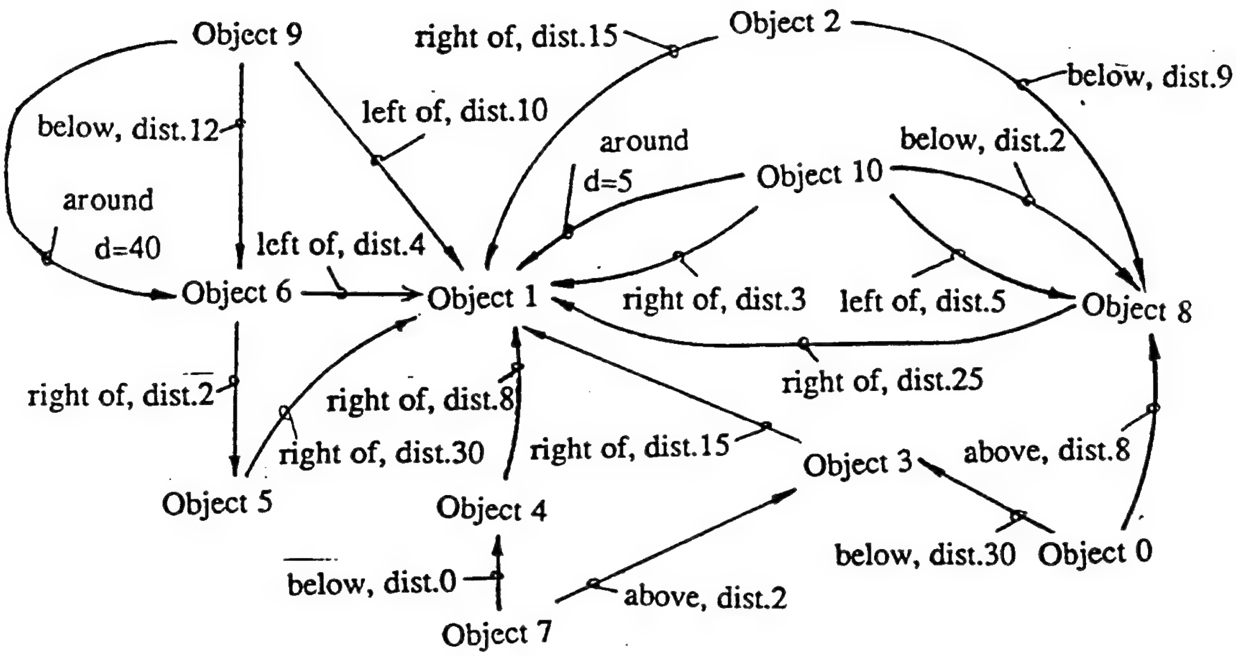


Figure 3: Location network for a aerial image.

lists containing some indicators and values in the next level (see Table 2), which is similar to the top level.

The frame structure used here has the following advantages: First, an object with some kind of variations can be recognized based on the slot matching because only the values of a few slots are changed and mismatched. For example, suppose the grey scales of a building changes due to lighting, the shape of its image (the main feature) is still unchanged. Second, the incomplete description of the object can be modified in terms of a set of examples.

Since the knowledge base is a relatively independent module with a fixed format, the modification and maintenance of the knowledge base do not have an effect on the other parts of the system.

Slots	Lists
Name	object name
Specialization	e.g. road, house, river, bridge
Shape type	e.g. line, long arc, circle, square, rectangle, rhombus
Shape parameters	e.g., the slant of the line or arc, radius of a circle, length of the side
Search record	e.g. has been searched? has been found? has been recorded in the knowledge based?
number of reference groups	
reference	number of reference objects names of the references location relation with the reference quantitative description of the location
procedures for computing parameters	
...	...

Table 1: A frame of an object.

4 APPLICATION OF THE REFERENCE LOCATION RELATIONS

The preprocessing of an image is described in [6]: segment the original image, eliminate tiny regions, and obtain features of each region. A location network is obtained from a pre-processed image. It is usually a complicated directed graph. If there is no directed loop in the graph, an object and all its recursive references objects form a tree. For each group of location relations, we start with a image-size array with all elements set to zero. Based on the reference location relations of each group, the regions where the object may exists are set to one, when all reference objects are taken into consideration, the search region determined by this group of reference relations is obtained.

With all the search regions based on each group of reference relations, the region to be searched for the object can be determined by taking the intersection of those search regions from all groups of reference relations in the previous step. The procedure for determining the region to be searched is an exhaustive search all over the reference relation tree.

Based on the information about the object to be searched, e.g. the shape and geometric parameters, an evaluation function is used to measure the consistency between the features of a model and the features of objects in the searching area. When the best match is found, an interpretation of the image will be generated for this image.

5 IMPLEMENTATION AND RESULTS

This section discusses the implementation and the results. The input image, Figure 4(a), is an aerial photograph of the Hai river. The input image is segmented

Objects	Shape type	Parameter
0	rhombus	7, 3
1	long arc	75
2	long arc	25
3	line	10
4	line	45
5	rhombus	64, 10
6	long arc	15
7	rectangle	15, 8
8	rectangle	16, 6
9	rhombus	15, 15
10	long arc	15

Table 2: The shape information of the objects.

into Figure 4(b), ad after eliminating small regions, Figure 4(c) is obtained. Reference relations are given in Figure 3. Geometric parameters are calculated and given in Table 2. The procedure of determining the searching area for object 9 is shown in Figure 5. The target object is found by matching with the geometric information of the objects within the final searching area.

6 MAINTENANCE OF THE KNOWLEDGE BASE

There can be thousands of location relations in a knowledge base for a complicated scene. A lot of knowledge can be put in the knowledge base during the learning procedure. The knowledge base has to be tested and maintained after adding new knowledge or modifying out-of-date knowledge in order to eliminate redundant knowledge and keep the knowledge base consistent. It is a difficult task to test the performance of the knowledge base manually. Some algorithms have been developed to solve the automatic testing problems here. In the next subsections, we will discuss some issues related to the maintenance of the knowledge base.

6.1 Check reference loop of a location network

Searching procedures are determined by the location relations among the objects. Whenever a target has some reference objects, the reference objects are searched first. However, if these direct reference objects refer to other objects, we have to search the later before searching the former. Whenever a direct or indirect reference is the target itself, the search procedure runs into an endless loop. Reference loops have to be resolved in advance.

For example, in Figure 6(a), the location relations related to the target A form a subnetwork in Figure 6(b), which has no directed loop. If a new location relation "D is below A" is added to this subnetwork, as shown in Figure 6(c), the resultant network

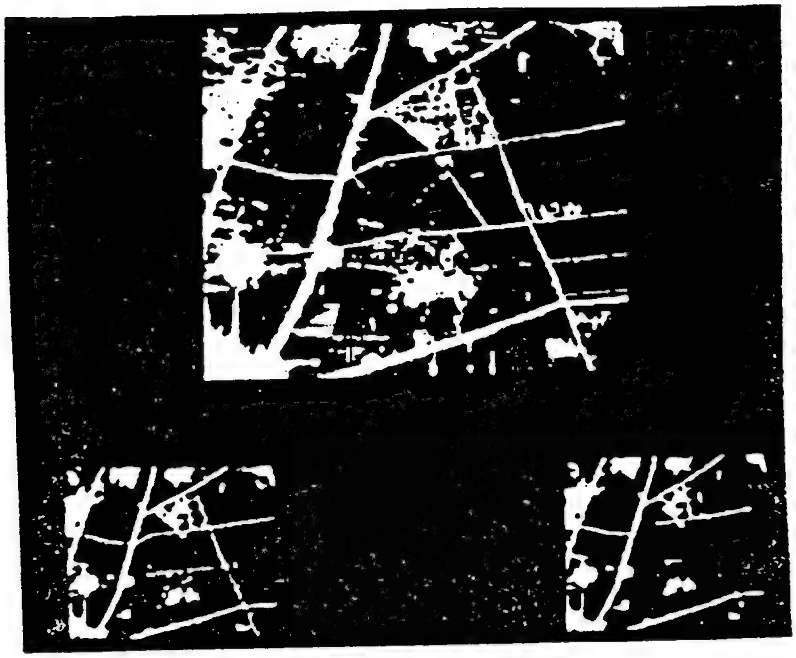


Figure 4: (a) The original image at the top; (b) The segmented image at lower left; (c) Obtained after eliminating small regions in (b).

contains a directed loop, therefore none of objects A, B and C can be searched based on it.

By omitting the group connectors and all the semantics associated with the edges from an original location network, a new directed graph of the reference relations is obtained. If there is a directed loop in this graph, there must be a loop of references. The loop can be detected using any exhaustive search method. Loops of references have to be broken by updating or eliminating some reference relations.

6.2 Check consistency of location relations

Consistency checking plays an important role in knowledge base maintenance since the facts and rules in the knowledge base must be consistent with each other. For example, if location relations "A is above B, A is below C, and B is above C" coexist in a knowledge base, such an object A can not be found. It is essential to check the consistency of locations before using them.

Consistency checking is to eliminate obvious logical inconsistency. Our knowledge base has relations such as above, below, left of, right of, inside, outside, mirror and around etc. The consistency of the three subset of location relations i.e. the subset of all "above, below", the subset of all "left-of, right-of" and the subset of all "inside, outside" is checked using simple logical inference. The checking procedures for all the three subsets are quite similar, therefore only the checking for "above, below" subset is discussed.

First, all directed edges with the semantics "above" in the original network are

(a)	(b)	(c)
(d)	(e)	(f)
	(b) g	

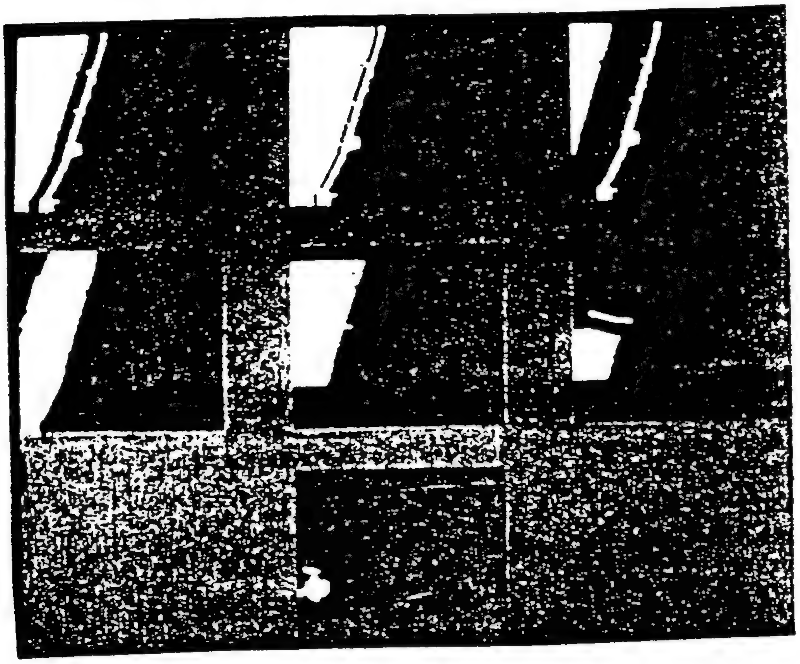


Figure 5: The search procedure for object 9.

(a) the search area obtained based on the location of object 1 and the location relation between objects 9 and 1, in order to search for object 9, it is necessary to search object 6 first; (b) the search area of object 6 obtained based on the location relation between objects 6 and 1, in order to search for object 6, it is necessary to search object 5; (c) the search area of object 5 obtained based on the location relation between objects 5 and 1; (d) the final search area of object 6 obtained by modifying (b) based on the location relation between objects 6 and 5; (e) the search area of object 9 obtained based on "the object 9 is around the objects 6"; (f) the final search area of object 9 obtained by modifying (e) based on "the object 9 is below object 6"; (g) this area is found by searching object 9 in (f).

extracted to be put in a new directed graph. Second, all edges with semantics "below" are put to directed graph with direction reversed. Third, all "colored" edges keep their original color. With these steps, each loop in this graph represents a set of location relations which can not be satisfied simultaneously. However, it is important to note that because of the "or" relation among the same colored edges, a directed loop containing the colored edges does not necessary mean an inconsistent subset of the location relations. Figure 6(d) shows such an example. Therefore when all the directed loops are found, the relations of the colored edges in each loop has to be checked to determine if there is any inconsistency.

Label all the colorless edges with "T" and the colored edges with either "T" or "F". If there exists a set of labels such that at least one of the colored edges in each colored group can be labeled "T", yet at least one can be labeled "F" in each directed loop, then no inconsistency is found. See Figure 6(e) and Figure 6(f). In order to

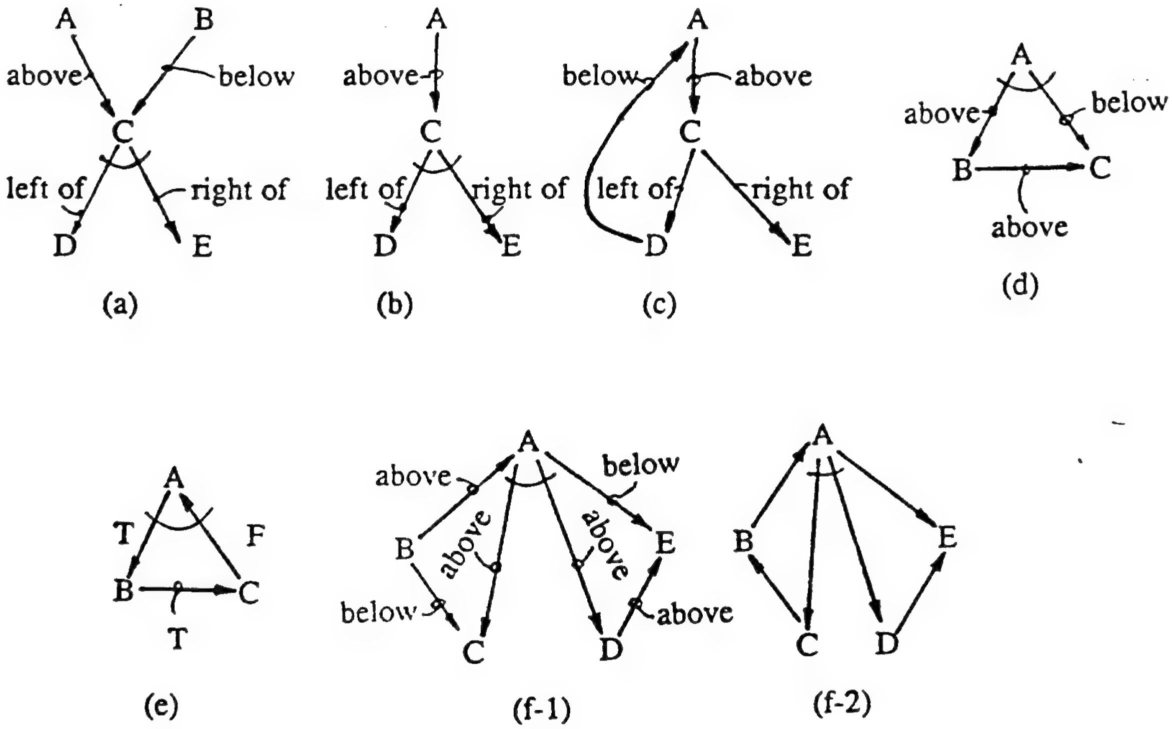


Figure 6: Some issues about knowledge base maintenance. (a) A location network built from the location reference relations. (b) the reference relation tree, during searching A based on the location network of (a). (c) A network with a reference relation cycle. (d) A consistent location network constructed by three relations "A is above B", "A is below C", and "B is above C" which can not be satisfied simultaneously. (e) the labeling of the location relation graph obtained from (d) shows that the location network is consistent. (f-1) an inconsistent location network. (f-2) a location relation graph from (f-1), there is no labeling scheme which can show the original network is consistent.

determine whether such a set of labels exists, a relation matrix of the loops and the colored edges can be used[8].

In fact, the procedure of finding the search area of the target is a procedure determining the consistency of the location relations. If a null search area is obtained after using some location relations, then these location relations are inconsistent, otherwise they are consistent.

6.3 Check redundancy of location relations

Redundant relations are do not affect final searching area, but increase computational cost. For example, if there are relations "A is above B, B is above C, and A is above C", then the last one is redundant and of no use for determining the search area of A.

Similar to consistency checking, all "above, below", all "left of, right of" and all "inside, outside" relations are put into three subgraphs and checked independently.

A redundant relation usually form a "D shape" graph with other relations. For example in Figure 7(g), there are two paths between nodes A and B: a direct path and an indirect path. These D-shape graph can be detected[8] using shortest-path algorithms of Warshall[7]. For example, Figure 7(h) is a D-shape graph where the relation "A is above B" is redundant. Figure 7(i) also consists of a D-shape graph, however, the relation "A is above B" is not necessarily redundant, since A–C and A–D are colored edges. Relation "A is above B" in Figure 7(j) is redundant.

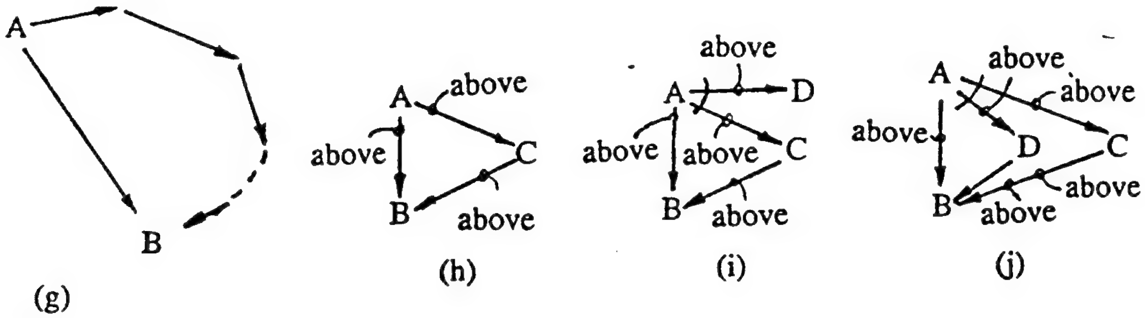


Figure 7: Some issues about knowledge base maintenance. (g) the necessary condition under which that the location relation between A and B is redundant: the directed edge A–B is in a D-shape graph. (h) a D-shape graph is formed by three location relations, where "A is above B" is redundant. (i) the relation "A is above B" is in a D-shape graph, but is not redundant. (j) there are two D-shape graphs in the location network, the relation "A is above B" is redundant.

First, we have to find the D-shape graphs, i.e., the necessary condition for redundant relations. For an n node directed graph G , an $n \times n$ array C associated with it is constructed. If its element $C_{ij} = 1$, then there exists a directed edge e_{ij} $(v_i \rightarrow v_j)$, otherwise $C_{ij} = 0$. In the following algorithm for all the elements C_{ij} in C, we check whether there exist simple directed paths $v_i \rightarrow v_k$ and $v_k \rightarrow v_j$ for $k = 1, 2, ..., n, \quad k \neq i, \quad k \neq j$. The simple directed path mentioned here denotes such path which does not pass through any nodes more then once (called path for short). If there are paths $v_i \rightarrow v_k$ and $v_k \rightarrow v_j$, then one can track from v_i to v_j through v_k, and we will put all the paths from v_i to v_j on record, and if $C_{ij} < 2$, then $C_{ij} \leftarrow C_{ij} + 2$. After this process, if $C_{ij} = 0$, there is no path from v_i to v_j in G; if $C_{ij} = 1$, besides a directed edge there are no other paths from v_i to v_j; if $C_{ij} = 2$, there is no directed edge (v_i, v_j) but a path from v_i to v_j; if $C_{ij} = 3$, there are a directed edge (v_i, v_j) and some other paths from v_i to v_j in G. The last one is the case which we want to find for the D-shape graph checking, because for $C_{ij} = 3$ the D-shape graph is constructed by combining each path from v_i to v_j and the directed edge (v_i, v_j).

Algorithm

Comments : As mentioned before, let $G = (V, E)$ be a connected (or unconnected), directed graph, where $V = \{v_i | i = 1, .., n\}$ is a set of nodes and $E = \{e_{ij} | i = 1, .., n; j = 1, .., n\}$ is a set of edges. The objects in the location network are represented by the nodes v_i in V, while the directed edge e_{ij} in E connecting a pair of nodes v_i, v_j in VN is used to expressed that v_j is a reference of v_i. C – a $n \times n$ array for recording the connections among the nodes. R_{ij}– a set of paths from v_i to v_j in G. Each element in R_{ij} is a path which is a ordered pair of the node labels (v_i, v_j).

```
Step 1
if $(v_i, v_j) \in E$,   $C_{ij} \leftarrow 1$,   $R_{ij} \leftarrow \{(i, j)\}$ ;
    if $(v_i, v_j) \notin E$,   $C_{ij} \leftarrow 0$,   $R_{ij} \leftarrow \emptyset$   $\forall i, j$,   $1 \le i, j \le n$ ;

Step 2
(1) $k \leftarrow 1$;
    (2) $i \leftarrow 1$;
    (3) if $i = k$, goto (8);
    (4) $j \leftarrow 1$;
    (5) if ($j = k$ or $j = i$), goto (7);
    (6) if ($C_{ik} > 0$ and $C_{kj} > 0$) do;
        (a) $U \leftarrow R_{ik}$,
        (b) if $U \neq \emptyset$, suppose $U = \{u, U_1\}, U \leftarrow U_1$, do:
            (i) $V \leftarrow R_{kj}$,
            (ii) if $V \neq \emptyset$, suppose $V = \{v, V_1\}, V \leftarrow V_1$, do:
        ($\alpha$)   $w \leftarrow u$
        ($\beta$)   suppose   $w = (l | w_1)$,   $w \leftarrow w_1$ do:
                ($\beta_1$) if $l = k$,
                $R_{ij} \leftarrow R_{ij} \cup append(u, v)$,
                if $C_{ij} < 2, C_{ij} \leftarrow C_{ij} + 2$, goto (ii)
                ($\beta_2$) if $l \notin v$, goto ($\beta$),
                if $l \in v$, goto (ii),
            (iii) if $V = \emptyset$, goto (b);
        (c) if $U = \emptyset$, goto (7);
    (7) $j \leftarrow j + 1$; if $j \le n$, then goto (5), else goto (8).
    (8) $i \leftarrow i + 1$; if $i \le n$, then goto (3), else goto (9).
    (9) $k \leftarrow k + 1$; if $k \le n$, then goto (2), else end.
```

Function *append*(u, v) in the algorithm forms a new list by linking the list v behind list u, it can be defined recursively as:

- $append((\ \ )x) = x$
- $append((x|x)y) = (x|append(xy))$

and $w = (l|w_1)$ is a list with its head l and tail w_1.

The following two conditions must be satisfied when the direct path in a D- shape graph is redundant:

1. Label all colorless edges with "T" and the colored edges with either "T" or "F". With at least one of the colored edges in each group is labeled "T", no matter how the colored edges are labeled, there always exists a D-shape graph whose edges are all labeled with "T".

2. In the "D" graph whose edges are all labeled with "T", its direct path and either the first or the last edge of the indirect path must have the same starting or ending point in the original location network.

If the redundant direct edge is colorless, then it can be removed. However, if it is colored, it should be kept to preserve the color feature.

7 CONCLUSION

A new knowledge-based scene analysis system has been developed for searching an object based on domain specific knowledge. The strategy used is straightforward. If an object is to be searched, the reference objects related to it are first searched recursively until the last object without any reference is found, and an target searching area can then be reduced based on the location of reference objects. Finally the desired object can be searched only within the area where it may appear and found by matching the geometric features of the objects inside the target searching area.

The maintenance of the knowledge base involves three procedures:

1. Check for loop references. If there are any, break them.

2. Check for conflicts of location relations. If there are any, modify them.

3. Detect redundant location relations. If there are any, delete them.

The experiments about some objects in the aerial image are conducted and the results are shown. The implementation of this system is discussed in detail. The techniques developed here are suitable for the case where the location relations among the objects are obvious but it is difficult to give a precise description of the desired object. The more distinct the objects are, the higher search efficiency the system may have. With the sophisticated maintenance techniques, there is a significant improvement in the performance of this system. This system may be used for more complex images.

References

[1] Binford, T.O., "Survey of Model-Based Image Analysis Systems", Intern. Journal of Robotics Research, Vol. 1, No. 1, 1982, pp.18-64.

[2] Nagao, M. and Matsuyama, T., *A Structural Analysis of Complex Aerial Photographs*, Plenum Press, New York (1980)

[3] Selfridge, P. G., "Reasoning about Success and Failure in Aerial Image Understanding", Ph. D. Thesis, University of Rochester, 1982.

[4] Nazif, A. M. and Levine, M. D., "Low level Image Segmentation: An Expert system", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 6, 1984, pp.555-577

[5] Mckeown, D. M. et al, "Rule-Based Interpretation of Aerial Imagery", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 7, 1985, pp.570-585

[6] Levine, M. D., *A Knowledge-based computer Vision System*, computer Vision system, Ed. by Hanson , A. R. and Riseman.

[7] Aho, A. V., Hopcroft, J. E. and Ullman, J. D. *The Design and Analysis of Computer Algorithm*, Addison-Wesley Pub. Co. (1974)

[8] Crystal J. Su, "A Knowledge-based Image Understanding System", Master Thesis, Tsinghua University, P.R.China, 1985

PROCESS FAULT DIAGNOSIS FOR A MAGNETIC LEVITATION CONTROL SYSTEM BASED ON AN ARTIFICIAL NEURAL NETWORK

Ching-Yu Tyan and Paul P. Wang

Fuzzy Logic Research Laboratory
Department of Electrical Engineering
Duke University, Box 90291
Durham, North Carolina 27708-0291
ctyan@ee.duke.edu, ppw@ee.duke.edu

Dennis R. Bahler

Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27695-8206
bahler@ncsu.edu

ABSTRACT

Fault diagnosis has become an issue of primary importance in modern process automation as it provides the prerequisites for the task of fault detection. The ability to detect the faults is essential to improve reliability and security of a complex control system. Parameter estimation methods, state observation schemes, statistical likelihood ratio tests, rule-based expert system reasoning, pattern recognition techniques, and artificial neural network approaches are the most common methodologies developed actively during recent years. In this paper, we describe a completed feasibility study demonstrating the merit of employing pattern recognition and an artificial neural network for learning and training using the observation of system state variables for a complex magnetic levitation vehicle (MLV) system. Analytical fault symptoms were obtained by system dynamics measurements and observation and the classification was carried out through an artificial neural network. The neural network was first taught the different fault situations through training patterns. After the network was trained, it can achieve an overall classification accuracy of 99.78% for a disturbance-free MLV model and 91.4% for a model with track disturbance irregularities.

1 INTRODUCTION

One of the most important goals of intelligent automatic control systems is to increase the reliability, availability, and safety of those systems. A complex automatic system can consist of hundreds of inter-dependent working elements which are individually subject to malfunction or failure. Total failure of the systems can cause unacceptable economic loss or hazards to personnel. Therefore, it is essential to provide on-line operating information by a scheme of observation and monitoring which detects faults as they occur, identifies the type of malfunction of faulty components, and compensates for the faults by appropriate actions and management to meet reliability and safety requirements so that the system can indeed continues to operate satisfactorily.

A number of useful techniques for dynamic fault diagnosis systems have been suggested in the literature. Model-based parameter estimation methods are discussed in [1, 2]. Faults appear as parameter or state changes caused by malfunctions of components. The parameter and state changes are determined using state observer techniques. Patton and Frank [3] described a detailed introduction of fault diagnosis in dynamic systems from the viewpoint of both theory and application. Patton presented a robust fault diagnosis system using eigenstructure assignment of observers [4] and optimal unknown input distribution matrix selection [5]. On the other hand, Jones and Corbin employed a band-limiting filter approach to fault detection [6]. Kitamura applied fault detection to nuclear reactors [7]. Kumamaru proposed statistical methods for fault diagnosis based on state-space and input-output models [8]. Walker studied a fault detection threshold determination using the theory of finite state Markov processes [9]. It is conceivable that stochastic modeling can be more realistically extended to include Semi-Marlcovian processes. However, in virtually all these methods the relationship between the model parameters and the physical coefficients needs to be unique and preferably known. In reality, unfortunately, This seldom is the case.

Rule-based expert systems have also been investigated very extensively for fault detection and diagnosis problems. Tzafestas designed a fault diagnosis expert system using knowledge-based artificial intelligence methodology [10]. Tyan and Wang implemented a rule-based fault diagnosis expert system for an aircraft flight control system [11]. Fault diagnosis using rule-based expert systems requires an extensive knowledge base and the accuracy of the diagnosis depends on the structure of the rules. Moreover, creating and updating a complete and detailed rulebase is usually a time-consuming task and much process design expertise is needed as well. In contrast, pattern recognition methods do not require an analytical model of the process, nor enormous database capacities, and thus provide a convenient approach to cope with fault diagnosis problems. This spirit of pattern recognition techniques is to solve the problem via essential “features”. Perhaps the most meaningful and significant “features” is nothing more than “state variables”. In modern control theory, as it is well known that the state variables represent a set of most compact, structurally speaking, information of a dynamic system.

In recent years, artificial neural networks have been successfully applied in pattern recognition and classification. Kanellopoulos built a classifier for remotely-sensed satellite images using multi-layer perceptron networks [12]. Sameer and Garg developed a labeled object identification system using multi-level neural networks [13]. Fault Diagnosis utilizing neural network techniques has become quite an active research area recently. Dietz and Kiech constructed a real time system for jet and rocket engine fault diagnosis [14]. Sorsa showed the use of perceptron networks in fault diagnosis for a heat exchanger-continuous stirred tank reactor system [15]. Himmelblau discussed the detection of faults in manufacturing electronic panels using neural networks [16].

In this paper, we study the possible fault symptoms occurring in a magnetic levitation vehicle system. The fault diagnosis monitor is governed by eigenstructure assignment of state estimator and MLV system control is accomplished using a state feedback controller. The method proceeds in three stages. First, the MLV system dynamic state variables in steady state are estimated making the use of the state observer estimation techniques and all the steady-state data are collected as training patterns. Then fault symptoms are defined analytically according to physical system features and a neural network fault classifier is designed that uses the back-propagation algorithm. Finally, classification accuracy is evaluated via independent testing patterns.

2 MAGNETIC LEVITATION VEHICLE SYSTEM DESCRIPTION

Beginning around the 1970s, because of operational and environmental drawbacks of wheel railway and air transport, considerable effort was expended to find new solutions for fast track ground transportation systems. Scientists and engineers in many countries tried to develop new noncontact magnetic levitation and guiding transport systems. A major goal of this effort was to achieve speed ranges that are unattainable for the wheel system by overcoming the disadvantages of the traditional wheel track system, such as point load, abrasion, noise, vibration, friction, propulsion and braking forces [17, 18]. In Germany, the development of a magnetic suspension ground transport system was sponsored by the Federal Ministry for Research and Technology within the scope of the Ground Transport System Technology program. Efficiency, profitability, low impact on the environment, considerable preservation of resources, and a high degree of safety are important development objectives. Using state-space analysis and aircraft technology, a train has been designed, built, and tested for operation at speeds as high as 400 km/hr.

Figure 1 shows the cross section of a MLV system. The track is a T-shaped concrete guideway. Electromagnets are distributed along the guideway and along the length of the train in matched pairs. The magnetic attraction of the vertically paired magnets balances the force of gravity and levitates the vehicle above the guideway. The horizontally paired magnets stabilize the vehicle against sideways forces. Forward propulsion is produced by *linear induction motor* action between train and guideway.

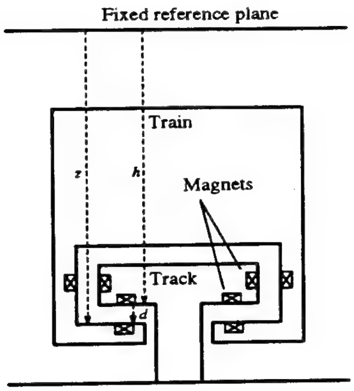


Figure 1: Cross section of a MLV train

2.1 SYSTEM DYNAMICS

The equations characterizing the train's vertical motion are now developed. It is desired to control the gap distance d within a close tolerance in normal operation of the train. The gap distance d between the track and the train magnets is

$$d = z - h$$

Then

$$\dot{d} = \dot{z} - \dot{h}, \quad \ddot{d} = \ddot{z} - \ddot{h}$$

where the dots denote time derivatives. The magnet produces a force that is dependent upon residual magnetism and upon the current passing through the magnetizing circuit. For small changes in the magnetizing current i and the gap distance d, that force is approximately

$$f_1 = -Gi + Hd$$

where G and H are positive constants. That force acts to accelerate the mass M of the train in a vertical direction, so

$$f_1 = M\ddot{z} = -Gi + Hd$$

For increased current, the distance z diminishes, reducing d as the vehicle is attracted to the guideway.

A network model for the magnetizing circuit is given in Figure 2. This circuit represents a generator driving a coil wrapped around the magnet on the vehicle. In this circuit

$$Ri + Li - \frac{LH}{G}\dot{d} = v$$

The three state variables $x_1 = d$ (gap distance), $x_2 = \dot{d}$ (gap velocity) and $x_3 = i$ (magnetizing current) are convenient, and in terms of them the vertical motion state equations are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{H}{G} & 0 & -\frac{G}{M} \\ 0 & \frac{H}{G} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ \frac{1}{L} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ f \end{bmatrix} \qquad (1)$$

where

$$u_1 = v \quad \text{(voltage control input)}$$

$$f = \ddot{h} \quad \text{(force disturbance of guideway irregularities)}$$

If the gap distance d is considered to be the system output, then the state variable output equation is $d = x_1$.

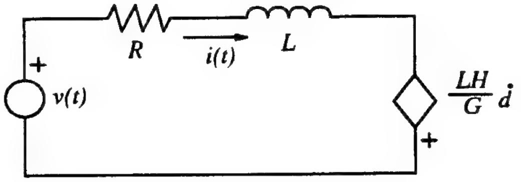


Figure 2: Magnetizing circuit model

The voltage v is considered to be a control input, while guideway irregularities $f = \ddot{h}$ constitute a disturbance. The system parameters M, G, L, and R can be derived analytically by static test and dynamic equilibrium of the vehicle. Suppose the train car weighs 8000 kg. Each car is supported by four magnets, each of which therefore supports $M = 2000$ kg. First a static test is performed. The air gap is clamped shut, causing d to be zero. A -120 V source is applied to the magnetizing circuit. With a time constant of $\frac{1}{30}$ of a second, -8 amps eventually flows at steady state. A resultant force of 4000 N is measured. Then the voltage is carefully varied until, at equilibrium, the car levitates with $d = 10$ mm under the influence of 8 amps of current. System parameters are computed as follows:

$$R = \frac{v}{i} = \frac{-120}{-8} = 15 \quad \Omega$$

$$L = RT = \frac{15}{30} = 0.5 \quad \text{H}$$

The data from when the air gap was clamped shut ($d = 0$) permits G to be computed

$$G = \frac{-f_1}{i} = \frac{-4000}{-8} = 500 \quad \text{N/Amp}$$

The data from when the car was levitated to equilibrium ($f_1 = 0$) permits H to be calculated ($H = 400$ N/mm). Therefore, the system state equation (Eq. 1) can be written as

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0.2 & 0 & -0.25 \\ 0 & 0.8 & -30 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ f \end{bmatrix}$$

2.2 POLE ASSIGNMENT DESIGN OF STATE FEEDBACK CONTROLLER & OBSERVER

The pole assignment design technique allows the assignment of the poles of the closed-loop transfer function to any desired location. Modern control theory introduces the concept of using system states to improve system performance based on state feedback. For some inaccessible states in a practical physical system, state observation provides a technique for estimating the states of a plant. Consider the case of a controllable and observable system governed by the state and output equations

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

The closed-loop system of state feedback and state observer can be represented by the composite form

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}-\mathbf{BK} & \mathbf{BK} \\ \mathbf{0} & \mathbf{A}-\mathbf{LC} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{e} \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \mathbf{u}_1 \qquad (2)$$

The characteristic polynomial for the matrix (Eq. 2) is

$$Q(\lambda) = |\lambda\mathbf{I} - (\mathbf{A}-\mathbf{BK})||\lambda\mathbf{I} - (\mathbf{A}-\mathbf{LC})|$$

Therefore, the eigenvalues of $\mathbf{A}-\mathbf{BK}$ and $\mathbf{A}-\mathbf{LC}$ can be assigned independently by the selection of appropriate matrix $\mathbf{K}$ and $\mathbf{L}$. This permits the controller and the observer to be designed separately. The overall system block diagram including state controller and observer is shown in Figure 3. If it is desired to have the system poles at $s = -1+j2, -1-j2$, and -3, then the desired characteristic equation is

$$\alpha_c(s) = s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0$$

$$= s^3 + 5s^2 + 11s + 15$$

and the matrix polynomial is formed using the coefficients of the desired characteristic equation

$$\alpha_c(\mathbf{A}) = \mathbf{A}^n + \alpha_{n-1}\mathbf{A}^{n-1} + \cdots + \alpha_1\mathbf{A} + \alpha_0\mathbf{I}$$

Then Ackermann's formula for the gain matrix $\mathbf{K}$ is given by

$$\mathbf{K} = [0 \quad 0 \quad \cdots 1][\mathbf{B} \quad \mathbf{AB} \quad \cdots \mathbf{A}^{n-1}\mathbf{B}]^{-1}\alpha_c(\mathbf{A})$$

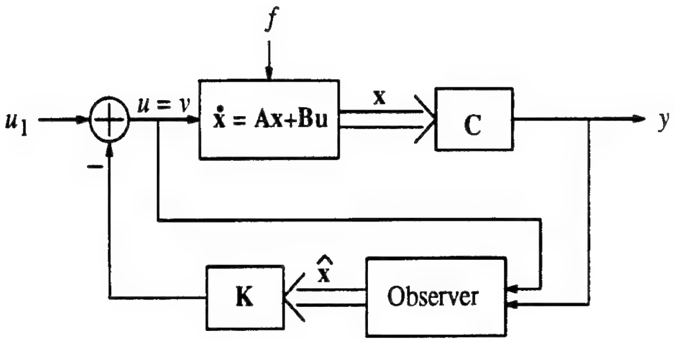


Figure 3: System block diagram with state-feedback and observer

$$= [k_1 \quad k_2 \quad k_3] = [12.5 \quad 22 \quad 32]$$

Similarly, suppose the observer poles are selected at $s = -10, -10, -10$, then the desired characteristic equation is

$$\alpha_o(s) = s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0$$
$$= s^3 + 30s^2 + 300s + 1000$$

and the matrix polynomial is formed using the coefficients of the desired characteristic equation

$$\alpha_o(\mathbf{A}) = \mathbf{A}^n + \alpha_{n-1}\mathbf{A}^{n-1} + \cdots + \alpha_1\mathbf{A} + \alpha_0\mathbf{I}$$

Then Ackermann's formula for the observer matrix **L** is given by

$$\mathbf{L} = \alpha_o(\mathbf{A}) \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 300 \\ 31976 \end{bmatrix}$$

The MLV system responses for each state variable with $u_1 = -300, f = 0$, and initial condition $x_0 = [0 \quad 0 \quad 8]$ are shown in Figures 4, 5, and 6, respectively.

3 ARTIFICIAL NEURAL NETWORKS

An artificial neural network can be viewed as a set of cells or neurons, each of which carries out a *sigmoid* type of computation, while receiving inputs from and sending outputs to other neurons. Although network models do not generally achieve human-like performance, they offer interesting means for pattern recognition and classification. The evolution of state $s_j(t)$ of neuron j receiving inputs from N neurons has the following behavior described by the *state transition equation*:

$$V_j(t) = \sum_{i=0}^{N-1} W(i,j)s_i(t) + I_j(t) - \theta_j$$

$$s_j(t + \Delta t) = f(V_j(t)), \quad j = 1, 2, \ldots, N$$

The coefficients $W(i,j)$ associated with the N inputs of a neuron j are the "synaptic weights." The neuron j is the postsynaptic neuron of the synapse associated with $W(i,j)$ and neuron i is its presynaptic neuron. The state of this neuron at time $t + \Delta t$ is affected by the states s_i rendered by the neurons i at time t and by an external input $I_j(t)$. The threshold θ_j of neuron j determines how "sensitive" it is to the inputs it receives. The potential $V_j(t)$ is found by adding the external input contribution $I_j(t)$ to the sum weighted by $W(i,j)$ of the state s_i presented to the inputs at time t, and subtracting the internal offset θ_j from the result. The new state $s_j(t + \Delta t)$ is calculated from the potential function through a nonlinear activation function f. In this study, a sigmoid function is used to insure that the activation function is differentiable.

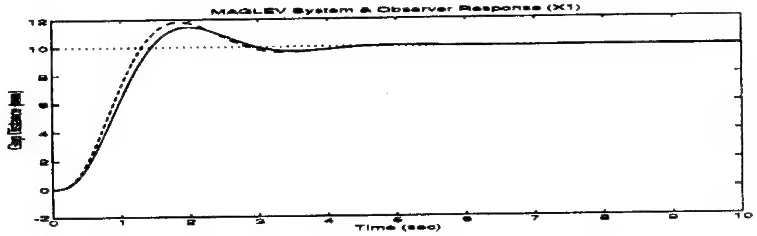


Figure 4: MLV response of state x_1

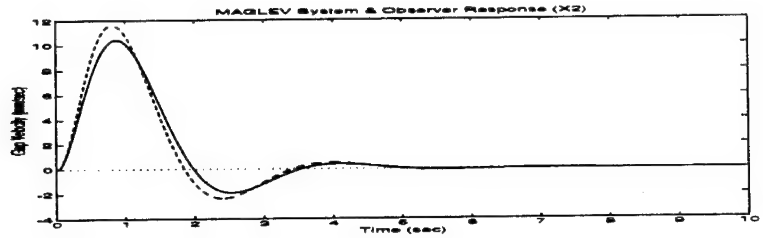


Figure 5: MLV response of state x_2

3.1 THE BACK-PROPAGATION ALGORITHM

Back-propagation was created by generalizing the Widrow-Hoff learning rule to multiple layer networks and nonlinear differentiable activation functions. Its learn-

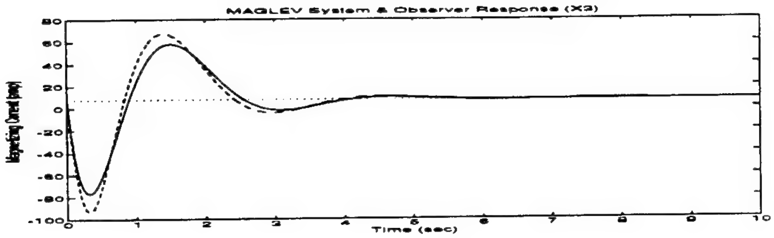


Figure 6: MLV response of state x_3

ing rules are used to adjust the weights and biases of networks so as to minimize the sum-squared error of the network. This is done by continually changing the values of the network weights and biases in the direction of steepest descent with respect to error. This is called a gradient descent procedure. During training, the network passes each input pattern through the hidden layer to the output layer to generate a result for each output node. It then compares the desired and actual results. The differences are the output-layer errors. Derivatives of error are calculated for the network's output layer, and then back-propagated through the network until the network's overall accuracy is improved by the aggregate corrections during the training. The general expressions for the back-propagation learning rule are

$$E(i) = T(i) - A(i)$$

$$\Delta W(i,j) = lr\Delta E(i)P(j)$$

$$\Delta B(i) = B(i) + lr\Delta E(i)$$

where
lr = learning rate, A = network output vector, T = target vector, P = input vector, E = error vector, and B = bias vector.

3.2 IMPLEMENTATION OF THE NEURAL NETWORK FAULT DIAGNOSIS CLASSIFIER

The chosen neural model (see Figure 7) for the MLV process fault diagnosis classifier is a fully-connected multilayer feed-forward network with sigmoid activation functions, trained by the back-propagation algorithm to minimize the sum-squared error. The input layer consists of 3 units encoding the steady-state values from each state variable. A choice of 20 hidden layer units gave the best network performance. The output layer consists of 10 units encoding a representation of 10 different fault classes. For example, the target vector for a fault belonging to class 3 would be $[0, 0, 1, 0, \cdots, 0]$. Nine representative fault situations are given in Table 1. The training data contain 70 patterns for normal operation and 70 patterns

No.	*Fault Situations*
1	Current leakage
2	Power source instability
3	Magnetic flux loss
4	Malfunction of vertical pair magnets
5	Malfunction of horizontal pair magnets
6	Vehicle overload
7	Motor burnout
8	Fouled tracking input set point
9	Power source breakdown

Table 1: Fault symptoms of MLV system

for each fault situation. Figure 8 presents the training data of 700 simulated observations for normal operation (label N) and all fault symptoms (label from 1 to 9) of a disturbance-free MLV system. Figure 9 shows the case of training data of an MLV model with track disturbance irregularities. The network's response to a given input is determined by the output unit having the highest activation state with a 10% confidence level. The following modifications were made in order to speed up the learning phase:

- We added momentum ($mc = 0.95$) in back-propagation to prevent the network from getting stuck in a shallow local minimum. The mathematical expression of back-propagation with momentum can be written as:

$$\Delta W(i,j) = mc\Delta W(i,j) + (1 - mc)lr\Delta E(i)P(j)$$

- An adaptive learning rate was applied to decrease the training time by keeping the learning reasonably high while insuring stable learning, i.e., if the new error exceeded the old error more than a predefined error ratio, the new weights, biases, outputs, and errors were discarded and the learning rate was decreased. Otherwise the new weights, etc., were kept if the new error was less than the old error and the learning rate was increased.

- We chose initial weights and biases by the method of Nguyen and Widrow rather than picking purely random values. This tends to lead to a satisfactory classification with fewer training epochs.

3.3 SIMULATION RESULTS

The neural network simulations were carried out on a DEC 5000 workstation. The appropriate selection of network architecture, the number of nodes in the

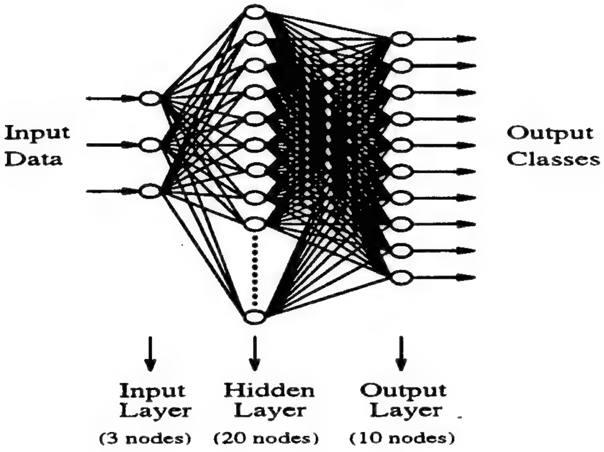
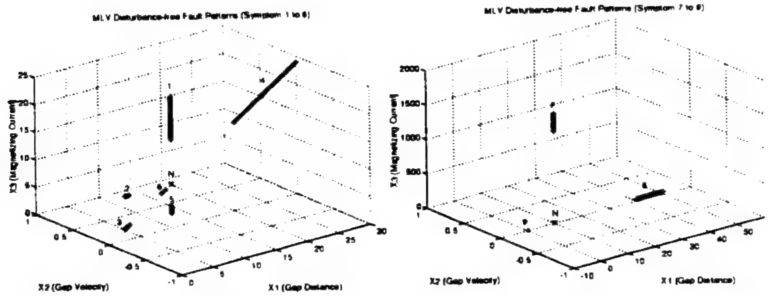


Figure 7: Network architecture of MLV system classifier

hidden layer, and the training parameters in the back-propagation algorithm required some experimentation. In the MLV system classifier example, a momentum value of 0.95, error ratio value of 1.04, and an adaptive learning rate value of 0.01 with an increase multiplier of 1.05 and a decrease multiplier of 0.7 were applied to speed up the training time. It was found that on the order of 14000 epochs were required to reach the system error goal using the training data of 700 observed measurement patterns. This took approximately 2 hours of cpu time on the DEC. Figure 10 shows the sum-squared error between the actual output and the desired output and the learning rate throughout the training period. Overall classification accuracies of 99.78% for the disturbance-free MLV model and 91.4% for the MLV model with track disturbance irregularities were achieved.

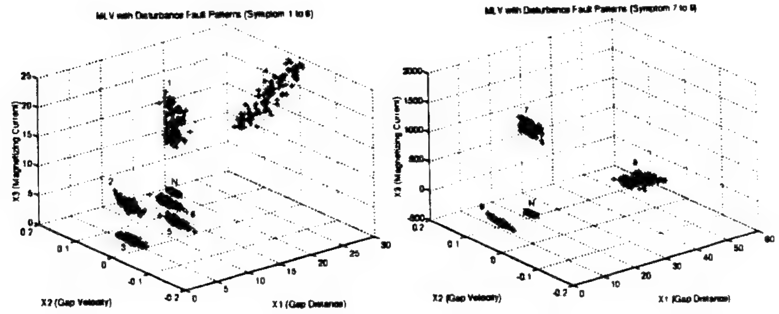
4 CONCLUSIONS

In this paper, a completed feasibility study of process fault diagnosis for a complex magnetic levitation vehicle control system using pattern recognition and artificial neural network classifier was presented. System performance was determined by a state-feedback controller and observed state measurement data in the steady-state were obtained via state estimator. The learning, training, and classification of system fault symptoms were carried out through an artificial neural network. It has been shown that a neural network classifier accomplishes a satisfactory classification accuracy in both disturbance-free and track disturbance irregularity cases. It is also important to note that the purpose of this paper is to demonstrate the concept of a "diagnostic doctor" for the dynamic systems. However, the dynamic system studied in this paper is a linear and time-invariant system. More difficult and complex nonlinear systems have also been investigated at present. Furthermore, teaching a neural network in general seems fairly slow; also, successful design of a neural network classifier is not an easy task. Moreover, if a new fault is introduced, network teaching must start all over again. All of these drawbacks will need to be overcome in order to make process fault diagnosis system more "intelligent." A full scale research effort is currently underway using the idea of "constraint processing" in process fault diagnosis systems and results are expected. A more thorough comparative study is nearly a certainty in the near future.



(a) (b)

Figure 8: MVL disturbance-free fault patterns (a) symptom 1 to 6 (b) symptom 7 to 9



(a) (b)

Figure 9: MVL with disturbance fault patterns (a) symptom 1 to 6 (b) symptom 7 to 9

5 ACKNOWLEDGMENT

This work was sponsored in part by the National Science Foundation under Grant No. ECS-9216474 and by the Electric Power Research Institute under Grant No. RP8030-3.

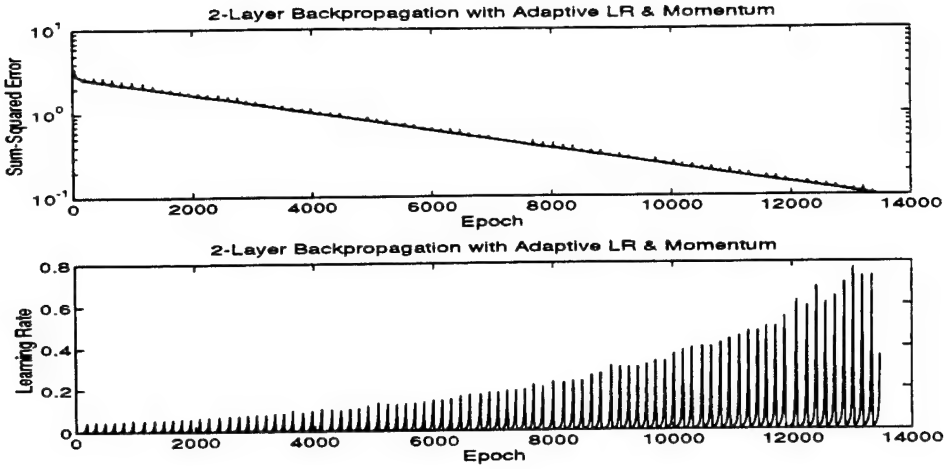


Figure 10: Neural network sum-squared error and learning rate during the training from 0 to 14000 epochs

References

[1] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, pp. 601–611, 1976.

[2] R. Isermann, "Process fault diagnosis based on dynamic models and parameter estimation methods," in *Fault Diagnosis in Dynamic Systems - Theory and Applications*, pp. 253–291, Prentice Hall International Ltd., 1989.

[3] R. Patton, P. Frank, and R. Clark, *Fault Diagnosis in Dynamic Systems - Theory and Application.* Englewood Cliff, NJ: Prentice-Hall, Inc., 1985.

[4] R. J. Patton and S. M. Kangethe, "Robust fault diagnosis using eigenstructure assignment of observers," in *Fault Diagnosis in Dynamic Systems - Theory and Applications*, pp. 99–154, Prentice Hall International Ltd., 1989.

[5] R. J. Patton and J. Chen, "Optimal unknown input distribution matrix selection in robust fault diagnosis," *Automatica*, vol. 29, no. 4, pp. 837–841, 1993.

[6] J. G. Jones and M. J. Corbin, "Band-limiting filter approach to fault detection," in *Fault Diagnosis in Dynamic Systems - Theory and Applications*, pp. 189–251, Prentice Hall International Ltd., 1989.

[7] M. Kitamura, "Fault diagnosis in nuclear reactors with the aid of parametric modelling methods," in *Fault Diagnosis in Dynamic Systems - Theory and Applications*, pp. 311–360, Prentice Hall International Ltd., 1989.

[8] K. Kumamaru, S. Sagara, and T. Soderstrom, "Some statistical methods for fault diagnosis for dynamical systems," in *Fault Diagnosis in Dynamic Systems - Theory and Applications*, pp. 439–475, Prentice Hall International Ltd., 1989.

[9] B. K. Walker, "Fault detection threshold determination using markov theory," in *Fault Diagnosis in Dynamic Systems - Theory and Applications*, pp. 477–508, Prentice Hall International Ltd., 1989.

[10] S. G. Tzafestas, "System fault diagnosis using the knowledge-based methodology," in *Fault Diagnosis in Dynamic Systems - Theory and Applications*, pp. 510–572, Prentice Hall International Ltd., 1989.

[11] C. Y. Tyan and P. P. Wang, "Rule-based fault diagnosis/detection expert system for aircraft flight control system," in *Proceedings of 2nd International Conf. on Fuzzy Theory & Technology*, pp. 17–21, 1993.

[12] I. Kanellopoulos, A. Varfis, G. G. Wilkinson, and J. Megier, "Classification of remotely-sensed satellite images using multi-layer perceptron networks," in *Proceedings of 1991 International Conf. on Artificial Neural Networks (ICANN-91)*, pp. 1067–1074, 1991.

[13] M. P. Sameer and D. P. Garg, "A labeled object identification system using multi-level neural networks," in *Proceedings of 2nd International Conf. on Fuzzy Theory & Technology*, pp. 204–207, 1993.

[14] W. E. Dietz, E. L. Kiech, and M. Ali, "Jet and rocket engine fault diagnosis in real time," *Journal of Neural Network Computing*, pp. 5–18, 1989.

[15] T. Sorsa, H. N. Koivo, and H. Koivisto, "Neural networks in process fault diagnosis," *IEEE Trans. on System, Man, and Cybernetics*, vol. 21, no. 4, pp. 815–825, 1991.

[16] D. M. Himmelblau, R. W. Barker, and W. Suewatanakul, "Fault classification with the aid of artificial neural network," in *Proceedings of IFAC/IMACS Symposium SAFEPROCESS'91*, pp. 369–373, 1991.

[17] K. H. Brock, E. Gottzein, J. Pfefferl, and E. Schneider, "Control aspects of a tracked magnetic levitation high speed test vehicle," *Automatica*, vol. 13, no. 3, pp. 205–223, 1977.

[18] K. Glatzel, G. Khrdok, and D. Rogg, "The development of the magnetically suspended transportation system in the federal republic of germany," *IEEE Trans. on Vehicular Technology*, vol. 29, pp. 3–17, February 1980.

AGENT-ORIENTED INFORMATION SYSTEM DESIGN

Egon M. Verharen, Hans Weigand

Infolab/Institute for Language Technology and Artificial Intelligence (ITK)
Tilburg University
P.O.Box 90153
5000 LE Tilburg
The Netherlands
tel. +31 13 662767 / 2806 fax. +31 13 663069 email: {egon,weigand}@kub.nl

ABSTRACT

In this paper a new method is introduced for the high-level specification of Information Systems. In this method, Information Systems are viewed as intelligent agents. The basis for the method is the separation of Environment of Discourse (EoD) and Universe of Discourse (UoD). Starting from organization and communication (authorization) modelling, a formal specification of application agents is derived. Combined with an object-oriented conceptual modelling approach for the UoD, a powerful method for the design of both database and interface of information systems is arrived at. Central in the proposed approach is the *contract* modelling concept and the specification of intelligent agents based on it.

The paper describes how the combined UoD/EoD model can be implemented in an agent-oriented system architecture by showing how contracts can be translated to the agent language Agent-0.

1. INTRODUCTION

The information system development process begins with an understanding of the organization under consideration and the data that constitutes its information infrastructure. To be most useful, information systems (IS) must be derived from this base of knowledge about the organization. [1] gives two requirements for ISs to be successful in business[1].

First the IS must have the ability to adapt to changing organization needs. Modern organizations must be able to adjust and adapt quickly to changes. They want ISs that support the business as it changes. The IS must be flexible and maintainable to quickly accommodate changes involving goals, products, markers, technology, governmental regulations and deregulations, and competitive shifts.

Second, the IS should incorporate accurate and consistent information. The information must have high integrity. It must not only be correct within acceptable precision, but consistent across the organization. For information to be properly interpreted and combined from all parts of the organization, a common model is needed.

The central task in IS design is building a Conceptual Model (CM) of the domain. Complying with the above mentioned requirements the CM is required to be complete

[1] Here the terms organization and business are used interchangably, since many of the requirements apply both to commercial businesses and other organizations.

(the 100% principle) so that it can provide the input to the implementation phase, and understandable to non-technical stakeholders involved. A third important requirement concerns reusability, extensibility and interoperability. *Reusability* means that parts of the specification -program code, data definitions, design objects- are effectively reused from one project to another. *Extensibility* means that the existing IS -both the implemented system and its design specification- can be easily extended and adapted following changes in the system requirements. *Interoperability* concerns the possibility to communicate with other ISs, on a technical and semantic level. Interoperable systems are autonomous rather than integrated systems whose behaviour is coupled to achieve a certain goal. In large organizations, one IS and one common CM is not attainable. Then interoperability and common CMs at the interfaces is the right level of aspiration.

In this paper we focus on the typical use of ISs in and between organizations to support the coordination of activities. This is achieved by *communication.* Communication, in our definition, is not just information, its essence is to commit agents to a course of action. Although not all activities can be fixed once and for all, due to uncertainties in the environment and the freedom of the subjects, it is nevertheless a constant organizational effort to explicate commitments in order to improve the cooperation and thereby get the job done. ISs are instrumental in this effort, not only by storing data and taking over certain routine tasks, but most importantly by explicating, up to the level of formalization, the rules of the communicative actions, the meaning of the terms and other kinds of mutual knowledge.

As mentioned above the Conceptual Model is an explicit description of the domain. Two aspects, or “projections”, can now be distinguished. The CM describes both the communication structures and the content of the communication. The former we represent in a so-called *Environment of Discourse (EoD)*, and the latter in the *Universe of Discourse (UoD)*. Typical objects in the EoD are the linguistic agents (human or computerized), the message types, and the rules that prescribe and describe the communication. The EoD describes the discourse itself as a process without going into the contents. What is said, and more in particular the meaning of these terms, is described in the UoD model.

The rest of this paper is organised as follows. In the next subsection a brief overview of our methodology for IS design is presented. In section 2 the Environment of Discourse and a technique for the modelling of this EoD are presented. Central in this section is the specification of *contracts*. Since our approach is strongly communication-driven we adopt an agent-oriented approach for specification of the contracts. The intelligent agents approach is described in section 3. Section 4 contains a comparison with other approaches. The paper is concluded in section 5 where also pointers to ongoing and further research are described. Focus in this paper is on the method and not on the formalisation of the models.

1.1. METHODOLOGY

In this section we present a brief overview of our methodology for Information System design [2]. Our methodology is strongly communication-driven and is based on the distinction between EoD and UoD. Here an organization (including the IS) is considered to be a system of acting and communicating agents. A description of the organization (EoD model) consists of the communication links between the agents and between them and (the agents in) the environment of the system. The part of the world to which the acting and communicating of the agents is related is called the object world and modelled by the UoD. The EoD/UoD distinction is essential.: it separates clearly the activities or operations of an organization from the things which are operated upon.

An agent is seen as an entity in the EoD with beliefs about the object world, an agenda (set of 'commitments', 'things to be done') and certain capabilities to perform acts, that communicates with other agents in the environment by sending messages.

The different phases in the development of an IS according to our methodology are represented in fig. 1. The different phases are represented as being sequential or parallel. Feedback between successive phases is possible but not represented in the figure.

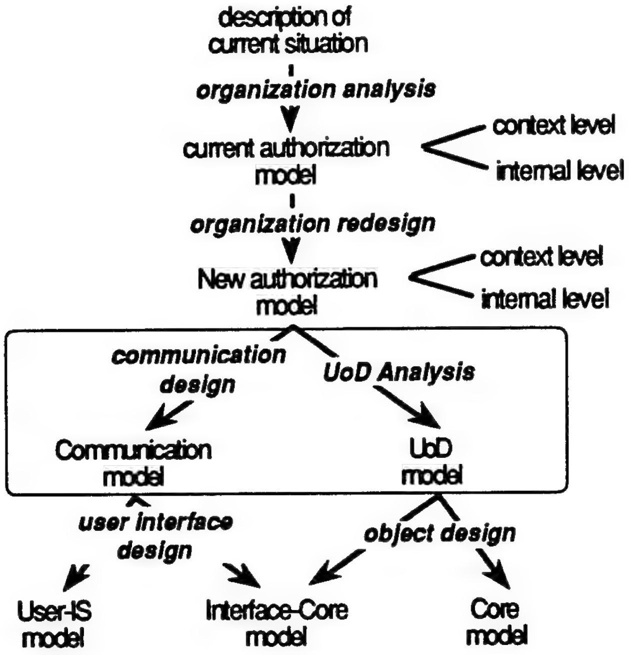


Fig. 1. Methodology for IS design.

Starting from a *description of the current situation* of the domain we can extract the *current authorization model*. This model describes the current obligations and permissions of the different agents (human as well as computerized) in the domain. An obligation is the result of a commitment to perform a certain act and permissions restrain or allow the commitment to and operation of an act. This model is usually the most difficult to get, since authorizations are often left implicit, but also the most crucial for the success of the IS in the organization.

Due to the introduction of the (new) IS(s) and other organizational changes the obligations and permissions may change. To model the impact of these changes we can look in the organization redesign phase at the communication between the environment and the IS, and changes in the communication between agents in the context. The new (desired) authorizations are modelled in the *new authorization model.*

Obligations and permissions belonging together logically can be grouped together in *contracts*. Therefore the authorization model takes the form of a set of authorization contracts. These contracts can be developed for the organization as a whole, but can also be developed bottom-up and per occasion.

The authorization contracts are very high-level specifications. They only describe the obligations and permissions of the different agents in the EoD with respect to communicative actions. Each contract may require several communication acts between the involved agents. In the communication design phase these communication acts are modelled in the *communication models*. The communication models are given by means of *communication contracts* (see section 2.1).

The authorization model(s) together with the communication model(s) give a semi-formal description of the behaviour and the communication interactions in the EoD. We call this the *EoD modelling*.

After the EoD modelling or in parallel, we model the UoD. The purpose of the*UoD modelling* is to derive a formal description of the content of the communication in the EoD, being the domain information. This description is called the *UoD model*. It is given in terms of objects and explicit relationships. Communication and interaction between objects is modelled by means of methods and triggers. This can be done by any OOD technique, such as the one described in [3]. Input for the UoD modelling comes from the authorization models as well as from the description of the current situation.

In this paper we do not describe the Interface modelling. The work is based on the separation of Interface and Core (application) as is common in Human-Computer Interaction research. See [4] for work on interface modelling.

Because we have the distinction between EoD and UoD we can distinguish between the behaviour and communications in the EoD and the behaviour and communications in the UoD. These two types of behaviour are quite different in nature and therefore require different modelling techniques. As already said above, the kinds of behaviour found in the EoD are authorizations and the interaction between agents following from these authorizations. The kinds of behaviour found in the UoD are mainly behaviour that is proper to the objects (part of their meaning) and requests for information or services. In the next section the modelling technique for the EoD is described.

2. EoD MODELLING

2.1.THE EoD MODEL - AN EXAMPLE

Typical elements of the EoD are the linguistic agents (human or computerized), the message types, and the rules that prescribe and describe the communication. By way of example, let us consider the well-known ISO case of car registration [5]. The excerpts of the case description we use for showing our ideas is:

"The universe of discourse to be described has to do with the registration of cars and is limited to the scope of interest of the registration authority. The registration authority exists for the purpose of:

1. knowing who is or was the registered owner of a car at any time from construction to destruction of the car.
2. to monitor certain laws, for example regarding fuel consumption of cars and their transfer of ownership

Manufacturers of cars:

There are a number of manufacturers, each with one unique name. Manufacturers may start operation, with the permission of the registration authority (which permission cannot be withdrawn). No more than five manufacturers may be in operation at any time. A manufacturer may cease to operate provided he owns no cars, in which case permission to operate lapses.

Cars:
A car is of a particular model and is given a serial no. by its manufacturer that is unique among the cars made by the manufacturer. The manufacturer is registered as the owner of the car as soon as practicable. At this time it is given one registration no., unique for all cars and for all time.

Fuel consumption:
Fuel consumption is a number of litres of hydrocarbon fuel per 100 kilometres, which will be between 4 and 25 litres. The fuel consumption averaged over all registered cars produced by a particular manufacturer in a particular year is required not to exceed a maximum value, which is the same for each manufacturer and may change from year to year. At the end of each January an appropriate message is sent by the registration authority to each manufacturer which has failed to meet this requirement.

"[5, ch. 4].

The EoD specification describes:

- the *agents* of the EoD, individual like `Registration Authority`, or generic, like `Manufacturer`;
- the *message types*, such as `operating request`, `granting permission`, `warning`, and `maximum fuel consumption declaration`;
- the *obligations* of the IS itself as one of the agents of the EoD; in this case the `Registration Authority`, for example (in structured form):

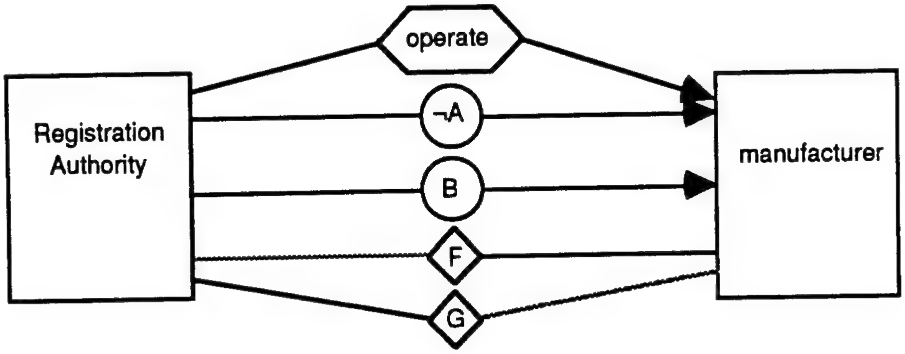
```
IF a manufacturer asks permission to operate
      AND the current number of operating manufacturers
      is not greater then five
THEN send_message(grant permission);
```

- obligations and permissions of the other EoD agents.

In our example the `Registration Authority` is the central agent in the EoD; others are the `Manufacturers` (with whom operating requests, warnings, etc. are exchanged) and the `Government`. The government is not mentioned in the description, but since the Registration Authority has certain responsibilities assigned, there must be an agent from which these assignments stem. For example, the `maximum fuel consumption` is supposedly declared by the government. Whether persons are agents in the EoD or not cannot be deduced from the case outline. In order to decide, we must know whether the persons have certain obligations or permissions towards the Registration Authority. Also the nature of certain rules has to be clarified, e.g., the rule that "a person cannot sell a car to a manufacturer". Does this mean that the Registration Authority should refuse such a car transfer if requested by the person, or that it should record such an illegal transfer (and inform the law)? This difference has to do with the question whether the communication between person and Registration Authority is *directive* (a request) or *assertive* (an information) (see below).

The example shows that building the EoD model is not a simple task, since we must be very clear about the permissions and obligations of the organizational agents. However, we consider this an *advantage*, because it is exactly one of the most important functions of conceptual modelling and because agreement between the partners on these structures is critical for the success of the subsequent phases of development.

The communication modelled in the EoD takes the form of speech acts between interacting subjects, and the relationships between these speech acts. For this reason, most traditional methods for dynamic modelling are not appropriate. For instance, *State Transition Diagrams* are not very useful since they focus on one object only and hide the interactions. *Data Flow Diagrams* (DFDs) make a distinction between processes and data. However, this distinction breaks down in the case of speech acts since these are actions and data flows at the same time. In [6], and [7] *Communication Structure Diagrams* (CSDs) are introduced for modelling the EoD. We present here a slightly modified version of CSDs. Fig. 2 gives an example of a CSD for the car registration case.



operate: The Reg.Auth. permits the manufacturer to operate
¬A: The manufacturer should not exceed max.fuel cons.
B: manufacturer should pay fine x
F: The manufacturer asserts the production of a car with model and serialno. in a certain year
G: The Registration Authority assigns the car a unique reg.no.

Fig 2. Example CSD from car registration case

The basic types of messages exchanged between agents in the EoD are based on the illocutionary primitives direct, commit, assert, declare and express [8]. A CSD depicts agents as rectangles and actions as links between them. A distinction is made between *directive speech acts* (circles), *assertive speech acts* (diamonds) and *material actions* (triangles).

In each case, we can distinguish two roles. In the example directive speech act, agent `Registration Authority` is the one that *requests* the performance of the action "not to exceed the maximum fuel consumption", whereas agent `Manufacturer` *promises* the same performance (for the success of the speech act a commitment to perform the action is made). In the example assertive speech act, subject `manufacturer` *produces* the message conveying the fact of the production of a car whereas subject `Registration Authority` *inspects* the same fact. For more details, see [6, 7, 9].

We have extended the modelling technique with *permissions* (hexagons), *conditions* (boxes), *triggers* (arrows) and *contracts*. (rounded boxes) Permissions are specific speech acts expressing and checking (the transfer of) authorization. Condition boxes are connected to the action links; depending on the position, they can be used to specify *preconditions* or *postconditions* of the action (in a logical calculus). Contracts provide a means to group related speech acts together and will be discussed shortly.

In the view of Dietz, CSDs model *essential* communications only, that is, unproductive speech acts are eliminated. For example, whether a confirmation of the acceptance of the `maximum fuel consumption` directive is sent or not cannot be deduced from the diagram. This principle lifts CSDs up to a much higher level of abstraction than traditional information flow diagrams. One should not confuse our approach to communication modelling with conversation modelling as used in natural language understanding research and natural language interface design. In our methodology, communication is modelled at two levels: first, we have the authorization model, and then, by refinement, the communication model. In conversation modelling, attention is given to factors such as feedback, turn-taking, things that the CSDs try to abstract from.

The *authorization model* corresponds most closely with Dietz' idea of essential communication. It describes the obligations and permissions of the subjects with respect to the speech acts, and the conditions attached to them, such as they are agreed upon by the organizational agents involved. This implies that a certain speech act is successful **iff** the speaker is authorized to that speech act according to the authorization model [10].

The *communication model* can be considered as a refinement of the authorization model. It describes *all* the message types that can occur in the EoD. For example, with respect to the assertive speech act link in the authorization model between `Manufacturer` and `Registration Authority`, it specifies a message type by which the `Registration Authority` can send a *query* to `Manufacturer`, or a message type by which the `Manufacturer` can (regularly) *inform* the `Registration Authority`.

2.2. CONTRACTS

Usually, some speech acts in the CSD belong logically together. For example, the ordering of parts, the delivery and the payment; or the notification of the production of a car and the assignment of a unique registration number make up one group. We propose to add the concept of *contract* to CSDs.

A 'contract' is a set of related authorizations (communications) together with conditions on the relationship between de acts and rules governing the violation of permissions and obligations.

The grouping of related authorizations in a contract has the following advantages:

- Contracts can be modelled as complex objects, and can be referenced as such. This allows both for very complex contracts to be split up in smaller parts and also to refer in a contract to another contract.
- Contracts can also be specialized and generalized. A contract is a specialization of another contract if it adds new authorizations and/or strengthens the pre- or postconditions. In this way, the contracts can be put in an inheritance hierarchy. More general contracts can be reused then from one application to another.

The agents in the EoD can be organised in a single specialization hierarchy as well. E.g., a mechanic IS-A employee IS-A person. In this hierarchy more specific agents inherit the authorizations from their more general parent(s). Agents can also be decomposed (aggregation structure). In this way, it is possible to specify authorizations top-down, starting at the company level and then going down to department or section level up to the individual agents.

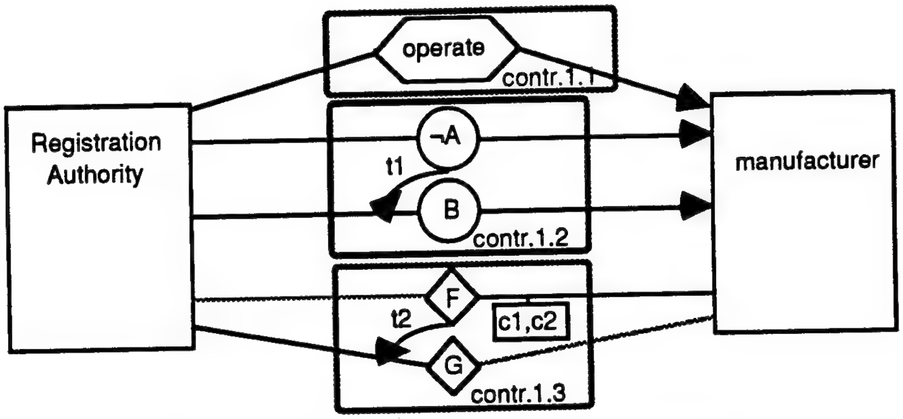
Note: The "agents" here can play what is usually called in conceptual modelling"roles". It is possible that one human or electronic agent acts as many agents. So when we refer to an agent in the EoD modelling we therefore usually describe only a "role" of that agent.

Since directives and assertives are actions, they can be defined by means of pre- and postconditions. A distinction is made between preparatory and procedural conditions.

Preparatory conditions are conditions that must be satisfied before the speaker is authorized to perform the speech act. If they are not met, the statement is void. *Procedural* conditions are conditions on the contents of the speech act. Both should be distinguished from *material* conditions, the preconditions of the action in question. For example, consider the registration of a new car. The assertive can be modelled as

`ASS(register(a:RA,x:car(d:model,s:serialno),m:manufacturer)(m,a)`

A preparatory condition for this assertive is that the manufacturer should have permission to operate. Otherwise, cars cannot be registered. A procedural one is that the `serialno` for the car is unique. The material conditions refer to the meaning of the action "register". For example, the precondition is that the car does not have a serial number, and the postcondition is that it has one.



operate: The Reg.Auth. permits the manufacturer to operate
¬A: The manufacturer should not exceed max.fuel cons.
B: manufacturer should pay fine x
F: The manufacturer asserts the production of a car with model and serialno. in a certain year
G: The Registration Authority assigns the car a unique reg.no.

contr.1.2:
t1: IF violate(¬A) THEN B

contr.1.3:
c.1: [PREP] PERM(operate, manufacturer)
c.2: [PROC] UNIQUE(serialno.)
t.2: IF F THEN G

Fig 3. Example contract model

Fig. 3 gives the simple CSD example from fig. 2 extended with contracts and some necessary conditions on the actions. The conditions are specified in a formal constraint language. The whole model can also be rendered in textual format (see fig.4, see [7] for the formal definition).

The *communication* model based on this contract may have additional messages, such as from `manufacturer` to the `Registration Authority` in which it complies to the directives (`not exceed max.fuel consumption` and `pay`); the contract shows the essential communication only.

```
Contr.1.2
 directive
    DIR(NOT(exceed_max_fuel_consumption(s,x,a)))(a,s)
 parameters
    x: car
    s: manufacturer
    a: registration authority
 preconditions
    [PREP] issued_max_fuel_consumption(government)
 sanctions
    pay(s,$y,a)
 end
 directive
    DIR(pay(s,x,a))(a,s)
 parameter
    x: amount
    s: manufacturer
    a: registration authority
 triggered by
    DIR(exceed_max_fuel_consumption(s,x,a))
 end
 //formal specification of associated UoD model
 [omitted]
end contract
```

Fig. 4 Example contract specification

2.3. EoD MODELLING AND DEONTIC LOGIC

The EoD model can be formalized in Dynamic Predicate Logic. The extension is that we include actions of the form `DIR(a)`, `ASS(f)`, `PERM(a)`, where `a` is an action and `f` a first-order formula. The effect of these actions is essentially given by the following axioms (ignoring the preconditions):

```
[DIR(a)]O(a)
[ASS(f)]f
[PERM(a)]P(a)
```

where `O(a)` and `P(a)` are deontic logic operators. (cf. [11]). The first axiom states that after giving the directive `a`, the obligation to do `a` is created. The second axiom says that asserting a fact causes the fact to be true. This holds in the EoD given the appropriate preconditions of such a speech act. The third axiom states that after giving the permission to do `a`, `a` is permitted (not forbidden). For more details we refer to [10].

3. AGENT SPECIFICATION

Since our modelling method is strongly communication-driven and leans at the use of contracts between two entities, we feel that the agent-oriented paradigm as proposed by [12] is a clean way to implement our contract specifications. An *agent* has a certain local autonomy and its behaviour is not predefined but based on commitments to other agents that are made either statically at design time or dynamically during execution.

3.1. AGENTS

Our definition of agents cohers to Shoham who defines an agent as "an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices and commitments" [12, p.52], all described in a precise fashion. As Shoham points out agents inform, request, offer, accept, reject, compete and assist one another, based on ideas directly borrowed from speech act theory.

Knowledge the agents should have, consists of:

- authorizations (giving some agents tasks, obligations, permissions);
- semantic knowledge (about the object types dealt with);
- general pragmatic knowledge (when the agent gets a question, it should somehow answer. Either a result, based on his knowledge (or the knowledge of others he knows of) or "don't know").

In our methodology, the first two kinds of knowledge are derived from the contracts, whereas the third kind is supposed to be built-in in the agent. It allows for the abstraction of communication details.

A generic agent interpreter executes the following life cycle: at regular intervals it processes the incoming messages, thereby updating its mental state (including its beliefs and commitments), and then executes its commitments, possibly resulting in another change of state.

Agent-0 is a language for specifying agent behaviour developped by Shoham. For our purpose the language has been extended with the following communicative actions:
`(PERMIT t a action) ≡ (INFORM t a permission(action))`
`(RECEIVE t a matter)` and `(SEND t a matter)` for material actions
and also a shorthand for the much used "request to inform":
`(ASK t a b fact) ≡ (REQUEST t b (IF (B, fact) (INFORM t+1 a fact)))`

Furthermore we adopt the possibility to define macros (although the actual implementation of Agent-0 does not) for those commitments and communicative acts that are stated in the contract.

In case the message type is not described in the contract (meaning there is no obligation to do so) the agent should be able to generate an answer to the (unspecified) message. For example, for the 'request to operate' from the `Manufacturer` to the `Registration Authority`:
`(manufacturer REQUEST permission(operate))`
an answer has to be generated, based on the current mental state of the agent. General pragmatic rules needed for this generation should be part of the agent's knowledge.

3.2. EXAMPLE AGENT SPECIFICATION

The example contract of fig. 4 can be implemented in Agent-0 as follows. First we give a textual description of the agent's beliefs, capabilities and commitment rules, and then the code of the Agent-0 program. The knowledge of the agent consists of knowledge of the object types dealt with (specified in the UoD model), authorization (specified in the contract model) and general pragmatic knowledge (built-in). Since in Agent-0 predicates are not declared, the first type of knowledge is largely implicit, but in a more sophisticated agent system, this kind of knowledge would be explicit. For running the agent program also knowledge about the various instances should be present (this can be extracted from the EoD and UoD populations). Furthermore, the agent should have knowledge of how to update its beliefs.

The agent specification of the `Registration Authority` agent program consists of:

initial beliefs

- the actual number of operating manufacturers *(from the UoD population)*
- the actual names of all operating manufacturers *(from the EoD/UoD population)*
- unique registration number *(from UoD model)*
- maximum fuel consumption for that year (issued by the government) *(from government contract or by communication)*

capabilities

- grant permission to operate to manufacturers *(from contract)*
- issue unique registration number to manufacturer *(from contract)*
- issue appropriate message if average fuel consumption of manufacturer > max. fuel consumption *(from contract)*
- update number and name of manufacturers *(internal)*
- update unique registration number *(internal)*
- evaluate arithmic expressions *(internal)*

commitment rules

- answer requests for information from manufacturers *(internal)*
- provide unique reg no. if manufacturer registers model and serial no. *(from contract)*
- provide permission to operate, if possible, if manufacturer asks for it *(from contract)*
- check average fuel consumption and issue appropriate message if not OK *(from contract)*

In Agent-0 code:

```
// Registration Authority Agent (RA) program
// initial belief
(now (no_of_manufacturers  no))
(now (permission  manufacturer)) *
(now (average_fuel_cons  manufacturer  amount))*
(now (free  reg_no))

//capabilities
    //macros
    (grant_permission manufacturer operate time) =>
            (PERMIT time+1 manufacturer operate)
    (check_fuel_cons) =>
            (ASK 31jan myself government (31jan (max_fuel_cons ?x)))
    (issue_reg_no  manufacturer  model  serialno  time) =>
            (IF  (NOT  (B  (time  (car model serialno))))
                    (INFORM  time+1  manufacturer  (time  (free  reg_no))))

((grant_permission  ?manufacturer  operate  ?time)  true)
```

```
((issue_reg_no ?manufacturer ?model ?serialno ?time) true)
((DO ?time (calculate_average_fuel_cons ?man))
            (B (?time (average_fuel_cons ?man ?amount))))
((DO ?time (update_reg_no ?reg_no))
            (B (?time (free ?reg_no))))
((DO ?time (update_manufacturer ?man ?no))
            (B (?time (no_of_manufacturers ?no)))
            (B (?time (permission ?man))))

//commitment rules
(COMMIT(?man REQUEST (IF (B,?p) (INFORM ?t ?man ?p)))
            true (?man (IF (B,?p) (INFORM ?t ?man ?p))))
(COMMIT(?manufacturer REQUEST (grant_permission ?man operate ?t))
            (AND (B (?t (no_of_manufacturers ?n))) (?n < 5))
            (myself (DO now+1 (update_manufacturer ?man ?n)))
            (?manufacturer (grant_permission ?man operate ?t)))
(COMMIT(?man.f. INFORM (issue_reg_no ?man ?model ?serialno ?t))
            (AND (B (?t (permission ?man)))
                      (B (?t (free ?reg_no))))
            (myself (DO now+1 (update_reg_no ?reg_no)))
            (?man.f. (issue_reg_no ?man ?model ?serialno ?t)))
(COMMIT(myself REQUEST (check_fuel_cons))
            (AND (B (31jan (average_fuel_cons ?man ?y))) (?y > ?x))
            (?man (INFORM 1 feb ?man appropriate_message)))
```

4. OTHER APPROACHES / RELEVANT RESEARCH

As mentioned in section 2 there are hardly any analysis methods that make the EoD explicit. An exception in the field of knowledge acquisition is [13], that describes ontology research for the sharing and reuse of knowledge bases. It separates content ontology (divided in task ontology and domain ontology), communication ontology and indexing ontology. The first two correspond to the UoD specification and the communications specification, respectively.

Recently the OO IS development method *Fusion* [14] has adopted both the view of the Environment modelling and the separation of interface and application as a result of this. First a model is developed which shows the system and its place in the problem domain. It recommends to make the GUI an external agent from the system. One of the steps is to determine what messages (are permitted, and their sequence) will be sent to the system and what its responses should be, by this defining the behavior of the system with respect to the GUI at a high level (cf. also [15]). Following this one can focus on designing the system or use Fusion on the external agents using a similar process.

Similar to our environmental authorization diagrams Wieringa in [16] makes use of context diagrams in the high level modelling of an IS in an organization. Unfortunately, here only one IS can be the object of study.

Methods for specification of speech acts are far and between. For instance [17] presents a language DR and a modelling technique PN for deontic rules, but these model procedures rather than communication structures. SAMPO [18] model communication using speech act theory, but CSDs offer a higher level of abstraction by concentrating on the essential communication.

5. CONCLUSION

The need for flexible IS, based on reusable and sharable information bases, for the communication and coordination in organizations led to the described methodology for IS development. Based on the separation of EoD (organizational environment/domain) and the UoD (data and rules in the domain) first the authorizations are modelled at a high level of abstraction. From this the communication models (given in CSD diagrams) can be derived, describing the essential communicative (speech) acts. The communication models are extended with the 'contract' modelling concept. Contracts are sets of related authorizations (communications) together with conditions on the relationship between de speech acts and rules governing the violation of permissions and obligations. A formal specification of these contracts is pointed at. An agent-oriented implementation fits well in this approach. The input for the agent specification phase are the specification of the contracts describing the speech acts and the UoD models of the contents of this communication. Examples of all major modelling concepts and specifications are given. The examples are from the well-known ISO car registration case.

We feel that this method contributes in the analysis of complex organizations for building IS, and has particular advantages for both the high-level specifications of interfaces and the design of interoperable systems (see below).

Besides the clean separation of Environment of Discourse and Universe of Discourse, and as a result the separation of interface and database giving higher reusability, the major advantage of this approach is the high-level specification of the IS behaviour. The specification can abstract from communication details that are built-in in the intelligent agents. From the perspective of agent-oriented programming, the contribution of this is that it offers an analysis and design method that can be of great help in designing agents.

5.1. FURTHER RESEARCH

The focus in this paper is on the contract and agent specification. Work must be done on the generation of interface objects. Furthermore, we are looking at ways to better model the other agents' knowledge, beliefs and capabilities, necessary for intelligent communication. So far attention was on the agent's own knowledge. Linder et al. [19] have given a formal semantics for the specification of an agents capabilities and beliefs.

Since the contracts can be very rich in permissions, commitments, obligations and model all types of speech act we are looking at extending the set of message types a language like Agent-0 understands.

Also work is done on the representation of general pragmatic knowledge, in order to improve the agent's answer capabilities. We are also looking at the representation of communication units larger than single messages (protocols, conversations).

Until now we have only looked at developing an IS in a restricted organizational environment. We feel however that this approach bears valuable insights for enterprise modelling or domain analysis for IS planning. Another application area is interoperable systems. The interface between two separate systems can be described by a contract between individual agents. Therefore this approach is valuable at specifying this interface [20]. The contract does not require any commitments from the agents about the internal architecture and processing, as long as they comply to the agreements. Often interoperable systems connect systems from different organizational entities. In such cases, the contracts also have organizational significance.

ACKNOWLEDGEMENTS

We thank Olga De Troyer for many fruitful discussions about the contract specification and UoD modelling. This work has partially been supported by the CEC under ESPRIT BRA WG 6071 ISCORE (IS - COrrectness and REusability).

REFERENCES

[1]. Steven H. Spewak and Steven C. Hill, *Enterprise Architecture Planning: Developing a blueprint for data, applications and technology*, QED Publ.Group, Wellesley, MA, USA (1993).

[2]. O.M.F. De Troyer, E. Verharen, H. Weigand, "Modelling information systems dynamics", Proc. of the Int.l ISCORE'93 WS on Inf. Syst.- Correctness and Reusability, Hannover, sept.'93, U. Lipeck and G. Koschorreck (eds.), Informatik-Berichte 01/93, Univ. Hannover, Hannover (1993).

[3]. O.M.F. De Troyer, "The OO-Binary Relationship Model: A truly Object-Oriented Conceptual Model", Proc. of CAISE'91, 3rd conf. on Advance IS Engineering, R. Andersen, J.A. Bubenko jr and A. Solvberg (eds.), Lecture Notes in Comp.Science 498, Springer-Verlag, Berlin (1991).

[4]. E. Verharen, H. Weigand and O. De Troyer, "Beyond Methods:Modelling information systems dynamics", to appear in Proc. of the Int.l ISCORE'94 WS on Inf. Syst.- Correctness and Reusability, Amsterdam, sept.'94, R. Wieringa (ed.), Free University, Amsterdam (1994).

[5]. *Concepts and terminology for the conceptual schema and the information base*, Report ISO/TC97/SC5-N695, J.J. v. Griethuysen (ed.), ISO, 1982.

[6]. J.L.G. Dietz, "A communication-oriented approach to conceptual modelling of information systems", Proc. of CAISE'90, 2nd conf. on Advanced Information Systems Engineering, B. Steinholz, A Sølvberg and L. Bergman (eds.), Lecture Notes in Comp. Science 136, Springer-Verlag, Berlin (1990).

[7]. H. Weigand, "Modelling deontic integrity constraints in communication", DEON'91, Proc. of the 1st Int.l. WS on Deontic Logic in Comp.Science, J.-J.Ch. Meyer and R.J. Wieringa (eds.), A'dam (1991).

[8]. J.R. Searle and D. Vanderveken, *Foundations of Illocutionary Logic*, Cambridge Univ. Press (1985).

[9]. J.L.G. Dietz, "Modelling communication in organizations", Linguistic Instruments in Knowledge Engineering, R. Meersman and R.P. v.d. Riet (eds.), North-Holland (1992).

[10]. H. Weigand, "The linguistic turn in information systems", Proc. of Collaborative Work, Social Communications and Information Systems, R. Stamper, P. Kerola, R. Lee and K. Lyytinen (eds.), Elsevier Science Publ, IFIP (1991).

[11]. R.J. Wieringa, J.-J.Ch. Meyer, and H. Weigand, "Specifying dynamic and deontic integrity constraints", *Data & Knowledge Engineering*, vol. 4, no. 2, p.157-191 (1989).

[12]. Y. Shoham, "Agent-oriented programming", *Artificial Intelligence 60*, p.51-92, Elsevier (1993).

[13]. R. Mizoguchi, "Knowledge Acquisition and ontology", Proc. of Int.l. conf. on Building and Sharing of very large-scale knowledge bases (KB&KS'93), JIPDEC, Tokyo (1993).

[14] D. Coleman et al, *Object-Oriented Development - The Fusion Method*, Prentice-Hall (1993).

[15]. I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard, *Object-Oriented Software Engineering: a use-case driven approach*, ACM Press, Addison-Wesley Publ. (1992).

[16] R. Wieringa, "A method for building and evaluating formal specifications of object-oriented conceptual models of database systems", Rapport IR-340, Free University, Amsterdam, dec. 1993.

[17]. R.M. Lee, "Bureaucracies as deontic systems", *Transactions on Office Information Systems*, vol. 6, no. 2, p.87-108 (1988).

[18]. E. Lehtinen and K. Lyytinen, "Action based model of information systems", Information Systems, vol. 12, no. 3, p.299-317 (1986).

[19] W. vd Hoek, B. van Linder, J.J.-Ch. Meyer, "A logic of capabilities", to appear in Proc. of LFCS'94 (also available as Rapport IR-330, Free University, Amsterdam, juli 1993).

[20]. H. Weigand, "Towards a design methodology for interoperable databases", Proc. IFIP WG 2.5 Conf. Semantics of Interoperable Database Systems (DS-5), Lorne Australia, (1992).

INTELLIGENT USER EXPLANATION OF QUERY RESULTS IN OBJECT-ORIENTED DATABASES

Suk-Chung Yoon
Dept. of Computer Science
Widener University
Chester, PA 19013

Il-Yeol Song
College of Information Studies
Drexel University
Philadelphia, PA 19104

ABSTRACT

In the near future, we believe that we will need much more sophisticated answer-finding schemes in an object-oriented database in order to satisfy the needs of truly intelligent information system. In this paper, we introduce a method to apply the intensional query processing techniques of deductive databases to object-oriented databases. So, we can generate intensional answers to represent answer-set abstractly for a given query in object-oriented databases.

Our approach consists of four steps: rule generation, pre-resolution, resolution, and post-resolution. In rule generation, we generate a set of deductive rules based on an object-oriented database schema. In pre-resolution, rule transformation is done to get unique intensional literals and extended term-restricted rules. In resolution, we identify rules that are potentially relevant to a query. In post-resolution, we find relevant resolvents as candidates for intensional answers among potentially relevant resolvents. We also uses the notion of potentially relevant resolvents and relevant resolvents to avoid generating certain meaningless intensional answers.

Keywords: Object-Oriented Database, Deductive Database,
Knowledge-Based Systems, Artificial Intelligence

1. INTRODUCTION

Object-oriented databases have received a great deal of attention over the past few years and have been the subject of intense research and development efforts. In the near future, we believe that we will need much more sophisticated answer-finding schemes in an object-oriented database in order to satisfy the needs of truly intelligent information system.

When processing a query in object-oriented databases, a set of database instances (*extensional answers*) are usually returned to users for an answer set. That is, object-oriented databases provide the answer set in the form of an enumeration of objects retrieved from the databases. In certain queries, we may not be interested in the set of objects that satisfy a given query in a particular database state. Instead, we may be interested in the conditions and the characteristics that objects must satisfy, in any state, to belong to the usual extensional answer of a query. Object-oriented databases do not support that capability. However, this situation can be different in a deductive database that supports complex reasoning which is not possible in object-oriented databases. When querying a deductive database, users can get not only the answer of a query as a set of facts but also the answer of a query as set of formulas(*intensional answers*). Deductive databases can generate a set of first order logic formulas as an answer set for a given query. That is, deductive databases deal with two different types of queries which can be distinguished by the kind of answer they are providing.

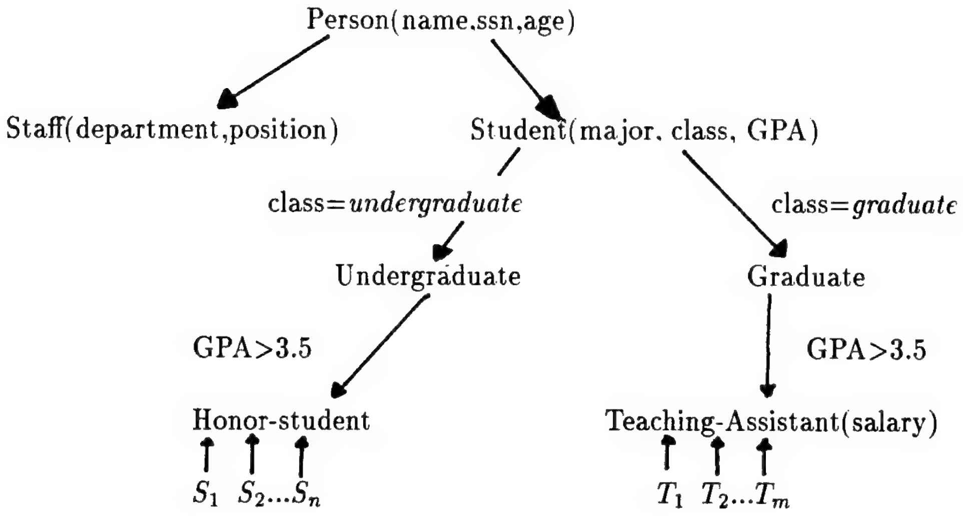
We can easily find many advantages in intensional query processing. As we get intensional answers as a set of formulas, they are independent of the particular circumstance in the database. Therefore, intensional answers provide a more stable answer than the extensional answers. Giving us exactly what conditions must be fulfilled to get a certain extensional answer, intensional answers provide a more compact and intuitive form than a set of facts could ever do. Intensional answers can be considered as a kind of interpretation or explanation of extensional answers. We can evaluate an intensional answer against the database and get a partial extensional answer. Intensional query processing has an advantage in computation compared with extensional answers because intensional answers can be computed without accessing the database, which greatly reduce the costs of processing a query.

In this paper, we introduce a method to apply the intensional query processing techniques of deductive databases to object-oriented databases. So, we can generate intensional answers to represent answer-set abstractly for a given query in obejct-oriented databases. In a great variety of ways to generate intensional answers, we are interested in an approach to construct resonably simple intensional answers. Each of them states a sufficient condition for a set of objects to belong to the extensional answer of the query. This paper is structured as follows. Section II introduces examples to show the advantages of intensional answers. Section III recalls basic definitions for object-oriented databases and deductive databases. Section IV surveys related works. Section V gives a precise definition for the concept of intensional answer. Section VI presents our approach to convert an object-oriented database schema based on class hierarchy into nonrecursive Horn clause notations and generate candidate intensional answers from logical consequences of nonrecursive Horn clauses and a given query. Section VII discusses possible extensions of our method.

2. MOTIVATING EXAMPLES

The following examples illustrate the advantages of intensional answers to a given query in an object-oriented database.

Suppose we have the following object-oriented database schema about a university.



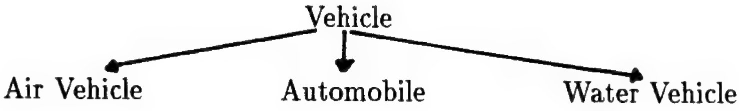
From the class hierarchy, we know that honor students are undergraduate students whose GPA is greater than 3.5 and that teaching assistants are graduate students whose GPA is greater than 3.5. The class Student contains information about students' names, their social security numbers, their ages, their majors, their classes(undergraduate or graduate) and their current GPAs. The first three attributes, name, ssn and age, are inherited from the super class Person.

Now, we suppose we have a query asking "names of all persons whose GPA is greater than 3.5". In our example, the extensional answer might be a long list of all honor students or all teaching assistants $=\{ S_1, S_2, ...S_n, T_1, T_2, ...T_m\}$. All these objects belong to different subclasses of the class Student. However, if a database system can process intensional queries, the answers can be represented as a set of simple classes such as

Ans_1 = { Honor-student: all honor students } and
Ans_2 = { Teaching-Assistant: all teaching assistants }

This example shows that extensional answers would only provide a list of all the objects that satisfy a query, whereas intensional answers distinguish between two cases for extensional answers. Thus, intensional answers are more informative than extensional ones. Intensional answers provide more insight into the nature of the extensional answer.

Suppose that there is an object-oriented database schema about vehicles and their types.



In the database, there is a class vehicle having the various vehicle types as subclasses. A user may query the structure of the database in a very natural way. For example, suppose the user wants to know what are all the vehicle types. In this case, if a database can support intensional answers, the answer can be represented as a set of simple subclasses such as Ans= {Air Vehicle, Automobile, Water Vehicle }. That kind of queries about the structure of the database can be handled easily.

3. OBJECT-ORIENTED DATABASES AND DEDUCTIVE DATABASES

We review the object-oriented data model and the deductive model, primarily derived from (Kim 1990, Gallaire 1984). Object-oriented databases and deductive databases have been areas of active research during the past several years. One common goal of these areas is to extend programming languages with database systems. The goal of objective-oriented databases is to apply object-oriented concepts in object-oriented programmings in modeling data. The goal of deductive databases is to integrate rules and traditional database of facts in logic programming to support complex reasoning.

Object-oriented databases have some features that do not exist in a conventional database. An object-oriented database is a set of object-oriented concepts, including object-identity, encapsulation, inheritance and polymorphism, for modeling data. These concepts are sufficiently powerful to support data-modeling requirements of many types of application. In object-oriented database systems, data are organized in a hierarchy of classes and subclasses. The class hierarchy captures the relationship between a class and its subclass. Subclasses inherit all the attributes of their superclasses and can have some of their own attributes. Object at any level of the hierarchy inherits all the properties of object higher up in the hierarchy. The access scope of a given query on a class, say C, is either the set of instances of C, or the set of instances of the entire class hierarchy rooted at C(that is, all subclasses of C). The result of the query may be the set of objects that belong to different classes within a class hierarchy.

Deductive databases have some features that do not exist in a conventional database. A deductive database is a database in which new facts may be derived from the facts that were explicitly stored by using an inference system. A deductive database comprises an extensional database(EDB) consisting of a set of facts explicitly stored in a physical database, and an intensional database(IDB) consisting of a set of deductive rules. These rules can be used to derive new facts from the facts in the EDB. In a deductive database, there is one advantage: facts, deduction rules and queries can be written in a uniform database language typically based on a first-order logic. A rule has the form(Prolog-like notation) $a : -b_1, b_2, \ldots b_m$ $m \geq 0$ where a is an atomic formula and b_i's are literals. All the variables occuring in the rule are assumed to be universally quantified. The literal a is called the *head* of the rule, the b_i's are referred to as the *body* of the rule. Every rule can be represented as a clause that is finite disjunction of one or more literals. . So, our rule can be represented as a clause, and would look like:

$$a \bigvee \neg b_1 \bigvee \neg b_2 \bigvee \ldots \bigvee \neg b_m$$

In this paper, we convert an object-oriented database schema into a set of non-recursive Horn clasuses. Horn clauses are a subset of clauses that have at most one positive literal.

A Horn clause is adequate for specifying a large class of database applications. An object-oriented database system can become a deductive object-oriented database system once it can directly support rules and various reasoning concepts. A deductive object-oriented database can get big improvements in productivity and functionality.

4. RESEARCH ON INTENSIONAL ANSWERS

Research related to intensional answers has received relatively little attention, compared to other research areas in deductive database systems. The common theme of research in this area is to provide all or part of the answers to a query in the form of formulas or abstract concepts. The relevant works for this paper are two different approaches.

Cholvy and Demolombe(1986) studied the idea of having a set of formulas as an answer set. They provided answers which were independent of these facts and valid in all the states associated with the set of facts. Their answers were a set of formulas defining the conditions that a given tuple must satisfy to be an answer for a given query. In their approach, only non-recursive rules were allowed to make sure that the algorithm for generating answers terminated. Based on the research by Cholvy and Demolombe, Pascual and Chovy(1988) developed an improved algorithm to retrieve an answer set of a given query. The former approach accepted any form of clauses and therefore is inefficient in the case of Horn clauses. The latter approach dealt with only Horn clauses to avoid numerous tests while generating the answer set. Thus, the latter approach accelerates the resolution process. However, the detailed steps to remove redundant resolution steps or meaningless answers were not discussed in Pasual and Cholvy(1988). These are important steps to reduce intensional query processing time. Song and Kim(1991) solved all of these problems left on Pasual and Cholvy's approach. They discussed a three-step intensional query processing scheme based on SLD resolution and discussed an implementation of an intensional query processor in Prolog. They use the notions of extended term-restricted rules, relevant literals and relevant clauses to avoid generating certain meaningless intensional answers. Yoon and Song(1994) generalized Song and Kim‘s approach and made their ideas efficient by using a graph structure.

Imielinski(1987), Motro(1989), and Pirotte, Roelants and Zimmanyi(1991) took different approaches from those discussed so far. Imielinsky introduced a new concept of an answer for a query. His answer can be composed of atomic facts and general rules built from projection, join and selection, so that they can be incorporated into the answer of a query. Pirotte et al.(1991) used integrity constraints to filter out inadequate answers. Motro(1989) also discussed the method that applies database constraints to generate intensional answers, and Motro and Yuan(1990) informaly discussed a query language incorporating intensional queries.

Our approach is applying Yoon and Song's work to object-oriented databases. Our approach deals with a non-recursive set of Horn clauses in order to get a terminating algorithm for generating the answers. Also, our approach avoids generating meaningless intensional answers.

5. DEFINITION OF INTENSIONAL AND EXTENSIONAL ANSWERS

We start this section by informally defining extensional answers and intensional answers as follows.

Definition: A *query* Q(X) in a deductive database is a formula where X is a tuple of free variables.

Definition: The *extensional answer* for a query is the set of tuples ā such that Q(ā) can be shown to be true on the extensional database when the deduction rules are taken into account.

Definition: An *intensional answer* to a query Q(X) is a formula A(X), obtained from the intensional database and from the query, that states a condition to be satisfied by tuples of values for x̄ in order to be part of the extensional answer of that query.

A formal definition of the intensional answers is given in Cholvy and Demolombe(1986). We define T as the database theory consisting of a set of facts and rules. Let Q(X) be a query. The intensional answer to a certain query Q(X), ANS(Q) is defined as:

ANS(Q) = $\{ ans_i(X) : T| - \forall X(ans_i(X) \longrightarrow Q(X)) \}$ (5.1)

The literal $ans_i(X)$ is defined so that under the theory T, any element X where X $\in$ ans_i(X), satisfy Q(X). That is, for any tuple of values a, if $T| - ans_i(a)$ then $T| - Q(a)$. ans_i(X) now can be any formula in a logical sense. As we are only interested in meaningful answers, which means answers within a defined domain of interest, we try to put some restrictions on the intensional answer set. Above all, this means that our intensional answers consist of only predicates from T. Also, we don't allow contradictory formulas or redundant answers.

So let DP= $\{P_1, \ldots, P_n\}$ be a set of predicate symbols either of the IDB or the EDB. Let L(DP) be the first-order language whose predicate symbols are $P_1, \ldots, P_n$. Thus, an intensioanl answer set ANS(Q,DP) to a query Q(X) is defined by:

ANS(Q,DP)= { $ans_i(X) : ans_i(X) \in L(DP)$ and
$T| - \forall X(ans_i(X) \longrightarrow Q(X))$ and
(ans_i(X) is not the negation of a tautology) and
(each $ans_i(X)$ is not redundant)} (5.2)

A redundant intensional answer is defined as follows: given two intensional answers: $ans_1(X)$ and $ans_2(X)$, $ans_1(X)$ is redundant if there is a following relationship between them $T| - \forall X(ans_1(X) \longrightarrow ans_2(X))$, that is, every tuple that satisfies $ans_1(X)$ also satisfies $ans_2(X)$, but not vice versa. A set of intensional answers is said to be complete if and only if the set of database values obtained from the set of intensional answers is equal to the set of database values in the extensional answer of the query. We need to add an additional condition to (5.2): ANS(Q,DP) must be complete. That is,

$$\forall \bar{a} \quad \Big(\bigvee_{v\text{-}ans_i \in \mathrm{ANS(Q,DP)}} v\text{-}ans_i(\bar{a})\Big) \equiv v\text{-}ans_E(Q)$$

where v-ans_i(ā) represents the set of database value tuples obtained from the set of an intensional answer and v-$ans_E(Q)$ represents the set of values obtained from the extensional answer.

If a query is formulated as a formula with free variable it is also possible to try to find bindings for these variables during the answering process. An extensional answer set then would be defined as:

$ans_E(Q) = \{X| - T \bigwedge[\forall X(Q(X) \longrightarrow ans_e(X))]\}$ (5.3)

An extensional answer is defined as a tuple ā which makes Q(ā) true in the database theroy T.

6. FORMALIZATION OF INTENSIONAL ANSWERS

In this section, we present the method of deriving intensional answers. We divide our approach into four phases: rule generation, pre-resolution, resolution and post-resolution. The four phases are partitioned into two categories: processing that can be done statically once and processing that has to be performed at run time. Rule generation and pre-resolution phases belong to the first category and the remaining two phases belong to the second category. So, the rule generation and the pre-resolution phases are independent of the queries posed to the database and hence are computed once prior to the processing of any query.

6.1 RULE GENERATION

In this phase, we perform three steps. In the first step, we generate a set of deductive rules based on the hierachy in an object-oriented database. We can classify rules into two different categories: rules to specify the class hierarchy and rules to specify relationship between two classes. The rules in the first category represent the hierarchy between a class and its superclass. For example, there is a class Vehicle having a vehicle type, say Airvehicle, as subclass. Then we have the rule: Airvehicle(x):- Vehicle(x). The rules in the second category represent the semantic knowledge between classes. For example, there are two classes, Automobile and Sportscar. If the number of cylinders in an automobile is greater than 6, then that automobile belongs to a subclass Sportscar. This semantic knowledge involoves two classes of Automobile and Sportscar. In this case, we many need to introduce a new literal called *property* literal. We can represent the above knowledge as follows:

Sportscar(x):- Automobile(x), Num-of-Cylinder(x,y), GT(y,6)

The literal Num-of-Cylinder is a property literal to show that the automobile x has y cylinders and the literal GT is a comparison literal to mean "greater than". After the first step, we get several rules that can be classified in one of two categories.

In the second step, we introduced EDB literals to represent the objects that belong to each class. For example, there is an object, say C1, that belongs to the class Car that has attributes such as id, price, size, num-of-door and gas-mileage. An object in the class Car can be represented in the EDB literal Car such as Car(10,25000,full,4,25).

In the third step, we remove literals that will cause recursion in rules. For example, suppose we have the following two rules:
Rule1: Automobile(x):- Bike(x)
Rule2: Bike(x):- Automobile(x), Num-of-Wheel(x,y), Less(y,3)
During resolution process, we may get another rules that will cause recursion as follows:
Bike(x):- Bike(x), Num-of-Wheel(x,y),Less(y,3)
So, we eliminate the literal Automobile in the Rule2. After this step, we will get a non-recursive set of rules. These tranformed rules are the essential components of the IDB.

6.2 PRE-RESOLUTION

In the first step some rule transformations will be done in order to get unique intensional literals and extended term-restricted rules.

Unique intensional literals mean that a literal should either be extensionally or intensionally defined but not both. One can always get rid of this equality of names by renaming the extensional literal to p^* and introducing the rule p:- p^*. The reason to force unique literals is to get a unique resolution tree.

Extended term-restricted rules have the following characteristics:

1. All the variables in the head of the rule appear also in the body
2. The rule does not have any constant in the head.

Converting our rules to extended term-restricted rules provides that all the information in the head of a rule also appears in the body. That keeps us from losing this information while doing resolution because the subsequent resolvent will also contain it. Using non extended term-restricted rules in resolution for the derivation of intensional answers will prevent the derivation of intensional answers for some queries, even though intensional answers exist(Song,1988).

6.3 RESOLUTION

Now we use SLD resolution with the notion of potentially relevant resolvents. One of the difficult problems in providing intensional answers is how to identify rules that are relevant to a query. Identifying relevant resolvents avoids getting meaningless answers. In our approach, we define a resolvent that is not potentially relevant as follows:

Rule 1: A resolvent is not potentially relevant of the resolvent (1) does not contain any intensional literal and (2) contains an extensional literal(s) and a property literal(s) and (3) the extensional literal(s) does not contain attribute about the property literla(s)(that is, the property literal(s) does not specify the attributes of the extensional literal(s)).

Rule 2: A resolvent is not potentially relevant if the resolvent (1) does not contain any intensional literal and (2) contains at least two property literals and/or two comparison literals that have same name and (3) factoring is impossible between those property literals and/or between those comparison literals. If factoring between two property or two comparison literals are possible, then we delete those two literals. After performing factorization, the resolvent may be empty.

Any resolvent which is not potentially relevant is not included in the subsequent resolution process, therefore avoiding the possibility of deriving meaningless intensional answers. Using the notion of relevant resolvents has two advantages over Cholvy and Demolombe(1986) and Pascual and Cholvy(1988). First, it can eliminate unnecessary rules which are used for the derivation of intensional answers. Second, certain meaningless intensional answers, as in Cholvy and Demolombe, can be avoided. The resolution stage serves to either simplify or discard resolvents. During this stage, users might be able to reduce the number of literals in a resolvent, or users might even be able to discard the resolvent itself totally.

6.4 POST-RESOLUTION

In this step, all the potentially relevant resolvents are taken to find a candidate for an intensional answer. Considering only the potentially relevant resolvents insted of all the resolvents ever generated in the whole resolution process is much more efficient and can

easily be justified. We use following rules to find relevant resolvents among potentially relevant rsolvents.
Rule 1: If there is a potentially relevant resolvent that is empty, we select the intensional literal as an intensional answer from the parent clauses participating in the resolvent. In this case, we use backtracking technique to find the intensional literal.
Ruel 2: If there is a potentially relevant resolvent that consists only extensional literals, find the intensional literal from the parent clauses participating in the resolvent and generate intensional answer by negating that intensional literal.

ALGORITHM

The following algorithm will compute intensional answers from a set of non-recursive Horn clauses consisting of EDB U IDB and a query Q(x). All these steps can be done in the order which the following algorithm presents.

1. Rule Generation

   - generate a set of deductive rules based on the class hierarchy and semantic knowledge in an object-oriented database.
   - introduce EDB literals to represent objects that belong to each class.
   - for each rule, remove literals that may cause recursion.

2. Pre-Resolution

   - make intensional literals unique by introducing new rules.
   - if necessary, transform rules into extended term-restricted rules.

3. Resolution

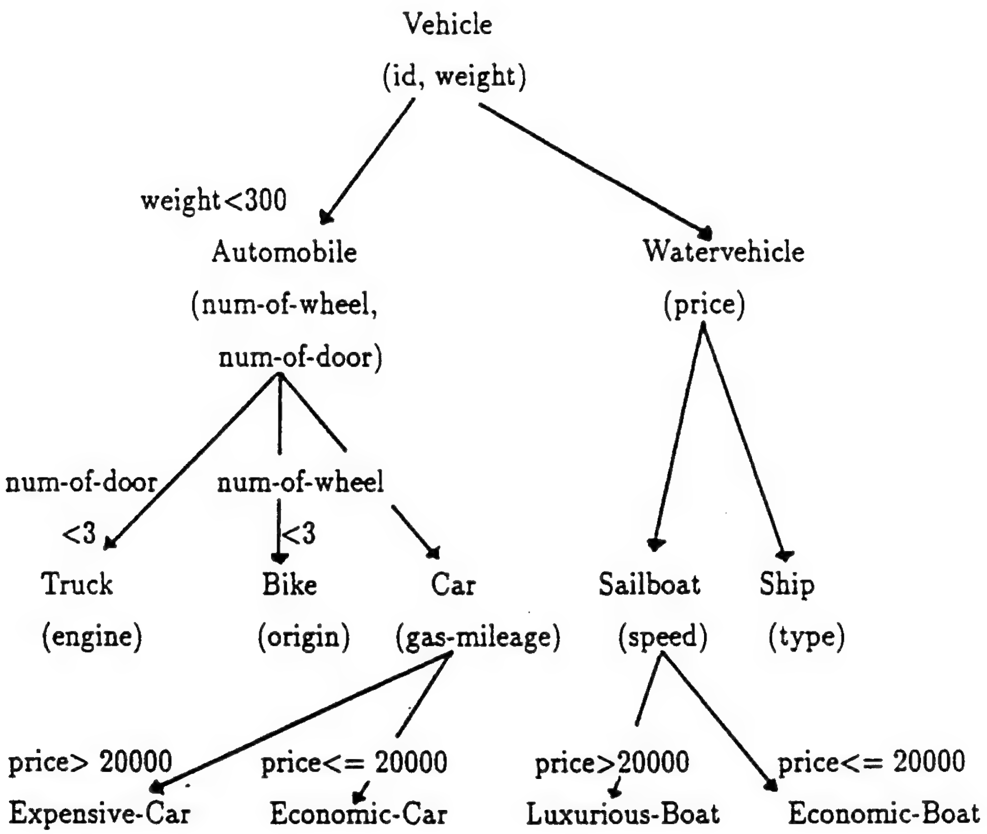
   - negate the query and convert it into the clause form
   - repeat
     - perform resolution
     - check if the resulting resolvent is potentially relevant according to rules in (6.3)
     - if no, then discard the current branch
   - until a resolvent consists only of non-intensional literals or it can not be further resolved.

4. Post-resolution

   - generate the candidates of intensional answers according to rules in (6.4)

AN EXAMPLE

This example shows the aplication of the algorithm introduced in the section above. Suppose we have the following class hierarchy about vehicles.



In the rule generation phase, we generate a set of deductive rules based on the above object-oriented database schema. In the first step, we get the following IDB rules that belong to two different categories.

IDB:
1. Rules to specify the class hierarchy:

Rule 1	Vehicle(x):- Airvehicle(x)
Rule 2	Vehicle(x):- Automobile(x)
Rule 3	Vehicle(x):- Watervehicle(x)
Rule 4	Airvehicle(x):- Plane(x)
Rule 5	Airvehicle(x):- Helicopter(x)
Rule 6	Automobile(x):- Truck(x)
Rule 7	Automobile(x):- Bike(x)
Rule 8	Automobile(x):- Car(x)
Rule 9	Watervehicle(x):- Sailboat(x)
Rule 10	Watervehicle(x):- Ship(x)

Rule 11 Car(x):- Expensive-Car(x)
Rule 12 Car(x):- Economic-Car(x)
Rule 13 Sailboat(x):- Luxurious-boat(x)
Rule 14 Sailboat(x):- Economic-boat(x)

2. Rules to specify properties between two classes:
Rule 15 Automobile(x):- Vehicle(x), Weight(x,y), LT(y,300)
Rule 16 Bike(x):- Automobile(x), Num-Of-Wheel(x,y), LT(y,3)
Rule 17 Truck(x):- Automobile(x), Num-Of-Door(x,y), LT(y,3)
Rule 18 Expensive-Car(x):- Car(x), Price(x), GT(y,20000)
Rule 19 Economic-Car(x):- Car(x), Price(x), LE(y,20000)
Rule 20 Luxurious-boat(x):- Watervehicle(x), Price(x,y), GT(y,20000)
Rule 21 Economic-boat(x):- Watervehicle(x), Price(x,y), LE(y,20000)

The comparison literal LT means "less than". The comparison literal LE means "less than and equal". The comparison literal GT means "greater than".

In the second step, we introduce the following EDB literals to represent objects that belong to each class.

EDB:
Vehicle(id, weight)
Airvehicle(id, weight, num-of-seat)
Plane(id, weight, num-of-seat, type)
Helicopter(id, weight, num-of-seat, speed)
Automobile(id, weight, num-of-wheel, num-of-door)
Watervehicle(id, weight, price)
Truck(id, weight, num-of-wheel, num-of-door, engine)
Bike(id, weight, num-of-wheel, origin, num-of-door)
Car(id, weight, price, num-of-wheel, num-of-door, gas-mileage)
Sailboat(id, weight, price, speed)
Ship(id, weight, price, speed)
Expensive-Car(id, weight, price, num-of-wheel, num-of-door, gas-mileage)
Economic-Car(id, weight, price, num-of-wheel, num-of-door, gas-mileage)
Luxurious-Boat(id, weight, price, speed)
Economic-Boat(id, weight, price, speed)

In the third step, we need to eliminate literals that will occur recursion. After that elimination, we will get the following transformed rules.

Rule 15 Automobile(x):- Weight(x,y), LT(y,300)
Rule 16 Bike(x):- Num-Of-Wheel(x,y), LT(y,3)
Rule 17 Truck(x):- Num-Of-Door(x,y), LT(y,3)
Rule 18 Expensive-Car(x):- Price(x,y), GT(y,20000)
Rule 19 Economic-Car(x):- Price(x,y), LE(y,20000)
Rule 20 Luxurious-boat(x):- Price(x,y), GT(y,20000)

Rule 21 Economic-boat(x):- Price(x,y), LE(y,20000)

In the pre-resolution phase, we need to rename the extensional literals to get unique intensional literals. Therefore, we add new rules to the IDB.
New rules:
Vehicle(x) :- *Vehicle**(x)
Airvehicle(x):- *Airvehicle**(x)
Automobile(x):- *Automobile**(x)
Watervehicle(x):- *Watervehicle**(x)
Truck(x):- *Truck**(x)
Car(x):- *Car**(x)
Expensive-Car(x):- Expensive-*Car**(x)
Economic-Car(x):- Economic-*Car**(x)
Luxurious-boat(x):- Luxurious-*boat**(x)
Economic-boat(x):- Economic-*boat**(x)
In this example, every rule is already in extended term-restricted form. We can skip the process to convert rules into extended term-restrcited rules. In the above rules, the predicates with the symbol *, which represents that those predicates are just EDB predicates, are used to simplify our process. For example, we use the predicate *vehicle** to represent the fact that the predicate *vehicle** means the EDB predicate *vehicle**(id,weight). Therefore, whenever we need to search EDB actually, we use the EDB predicate *vehicle**(id,weight).

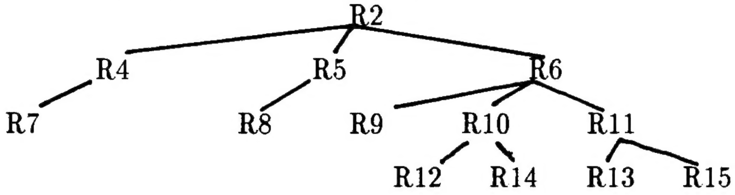
In the resolution phase, we use the SLD resolution with a query and the IDB and EDB gained from the previous phase. The query is negated and converted into the clause form. Suppose that we have the following query "find all vehicles whose price is greater than 20000"

:- Vehicle(x), Price(x,y), GT(y,20000)

After the first step, we get the following three resolvents:
R1) :- Airvehicle(x), Price(x,y), GT(y,20000)
R2) :- Automobile(x), Price(x,y), GT(y,20000)
R3) :- Watervehicle(x), Price(x,y), GT(y,20000)
For our convenience, we will only consider R2 branch.(We can use same technique for R1 and R3 branches.) After the second step, we get the following resolvents:
R4) :- Truck(x), Price(x,y), GT(y,20000)
R5) :- Bike(x), Price(x,y), GT(y,20000)
R6) :- Car(x), Price(x,y), GT(y,20000)
After the third step, we get the following resolvents:
R7) :- *Truck**(x), Price(x,y), GT(y,20000)
R8) :- *Bike**(x), Price(x,y), GT(y,20000)
R9) :- *Car**(x), Price(x,y), GT(y,20000)
R10):- Expensive-car(x), Price(x,y), GT(x,20000)
R11):- Economic-car(x), Price(x,y), GT(y,20000)
According to rule 1 in section 5.3, the resolvents 7 and 8 are not potentially relevant. We ignore those two resolvents. The resolvent 9 is potentially relevant. After the fourth step, we get the following resolvents:
R12) :- Expensive-*car**(x), Price(x,y), GT(y,20000)
R13) :- Economic-*car**(x), Price(x,y), GT(y,20000)

R14) :- Price(x,y), GT(y,20000), Price(x,y), GT(y,20000)
R15) :- Price(x,y), GT(y,20000), Price(x,y), LE(y,20000)
According to rule 2 in section 5.3, the resolvent 15 is not potentially relevant. After factorization, clause 14 is empty.

After the resolution phase, we have four potentially relevant resolvents R9, R12, R13 and R14. The resolution tree is as follows:



In the post-resolution phase, we apply rules in section 5.4 to those four potentially relevant resolvents to find relevant resolvents for intensional answers. We have an intensional answer from R14, Expensive-car(x). Similarly, we have an intensional answer from the R3 branch, Luxurious-boat(x).

7. CONCLUSION

Object-oriented database have received a great deal of attention over the past few years and have been the subject of intense research and development efforts.

In this paper, we have presented a method that addresses the important issue of providing more meaningful answers(intensional answers) to queries in an object-oriented database by using SLD-resolution technique. We apply the intensional query processing techniques of deductive databases to object-oriented databases. So, we can generate intensional answers in object-oriented databases. An intensional answer characterizes the extensional answer and provides more insight into the nature of the extensional answer.

Our approach consists of four steps: rule generation, pre-resolution, resolution, and post-resolution. In rule generation, we generate a set of deductive rules based on an object-oriented database schema. In pre-resolution, rule transformation are done to get unique intensional literals and extended term-restricted rules. In resolution, we identify rules that are potentially relevant to a query. In post-resolution, we find relevant resolvents as candidates for intensional answers among potentially relevant resolvents. We also uses the notion of potentially relevant resolvents and relevant resolvents to avoid generating certain meaningless intensional answers.

Our method is complete because it discovers all of the intensional answers. We avoid intensional answer that have little or no value to the user.

It may be interesting to use different computational methodologies to get intensional answers. Another method besides resolution might help to get intensional answers.

REFERENCES

1. Chang, C. and Lee, R. (1973), *Symbolic Logic and Mechanical Theorem Proving*, Academic Press

2. Cholvy, L. and Demolombe, R. (1986), "Querying a Rule Base", In *Proceedings of the First International Conference on Expert Database Systems*, pp. 365-371

3. Gallaire, H., Minker, J. and Nicolas, J. (1984), "Logic and Databases: A Deductive Approach", Computing Survey 16(2), pp.153-185

4. Imielinski, T.(1987), "Intelligent Query Answering in Rule Based Systems", Journal of Logic Programming, Vol. 4, No. 3, pp.229-258

5. Kim, Won (1990), *Introduction to Object-Oriented Databases*, MIT Press

6. Motro A.(1989), "Using Integrity Constraints to Provide Intensional Answers to Relational Queries", In *Proceedings of 15th VLDB Conference*, pp. 237-246

7. Motro A. and Yuan Q. (1990), "Querying Database Knowledge", In *Proceedings of the International Conference on Management of Data*, pp. 173-183

8. Pascual, E. and Cholvy, L. (1988), "Answering Queries Addressed to The Rule Base of a Deductive Database", In *Procceedings of the Second International Conference on Uncertainty in Knowledge-based Systems*, pp. 138-145

9. Pirotte, A. and Roelants, D. (1989), "Constraints for improving the generation of intensional answers in a deductive database", In *Proceedings of 5th International Conference on Data Engineering*, pp.652-659

10. Pirotte, A., Roelants D., and Zimanyi, E (1991), "Controlled Generation of Intensional Answers", In IEEE Transactions on Knowledge and Data Engineering, Vol.3, No. 2, pp. 221-236

11. Song, I.Y. and Kim, H. J. (1991), "Design and Implementation of a Three-Step Intensional Query Processing SCheeme", Journal of Data Administration, Vol. 2, No, 2, pp. 23-25

12. Song, I. Y. and Dubin, D. (1991), "Intensional Query Processor in Prolog", In *Proceedings of ISMM Int'l Symposium on Computer Applications in Design, Simulation and Analysis*, pp. 204-207

13. Ullman, J. (1988), *Principles of Database and Knowledege-base Systems*, Computer Science Press

14. Yoon, S.C. and Song, I.Y. (1994), "A General Method for Generating Intensional Answers in am Intelligent Information System", In *Proceedings of the 1994 ISCA International Conference on Computers and Their Applications*, pp. 94-98

15. Zdonik, S. and Maier, D. (1990), *Readings in object-oriented database systems*, Morgan Kaufmann Publisher

OFFICE OF THE UNDER SECRETARY OF DEFENSE (ACQUISITION)
DEFENSE TECHNICAL INFORMATION CENTER
CAMERON STATION
ALEXANDRIA, VIRGINIA 22304-6145

IN REPLY REFER TO DTIC-OCC

SUBJECT: Distribution Statements on Technical Documents

TO:
OFFICE OF NAVAL RESEARCH
CORPORATE PROGRAMS DIVISION
ONR 353
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660

1995 1027 008

1. Reference: DoD Directive 5230.24, Distribution Statements on Technical Documents, 18 Mar 87.

2. The Defense Technical Information Center received the enclosed report (referenced below) which is not marked in accordance with the above reference.

PROCEEDINGS SUMMARY
N00014-94-1-0799
TITLE: PROCEEDINGS OF THE
EIGHT INTERNATIONAL
SYMPOSIUM ON METHODOLOGIES
FOR INTELLIGENT SYSTEMS

3. We request the appropriate distribution statement be assigned and the report returned to DTIC within 5 working days.

4. Approved distribution statements are listed on the reverse of this letter. If you have any questions regarding these statements, call DTIC's Cataloging Branch, (703) 274-6837.

FOR THE ADMINISTRATOR:

1 Encl

GOPALAKRISHNAN NAIR
Chief, Cataloging Branch

DISTRIBUTION STATEMENT A:

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

DISTRIBUTION STATEMENT B:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (Indicate Reason and Date Below). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT C:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTORS; (Indicate Reason and Date Below). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT D:

DISTRIBUTION AUTHORIZED TO DOD AND U.S. DOD CONTRACTORS ONLY; (Indicate Reason and Date Below). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT E:

DISTRIBUTION AUTHORIZED TO DOD COMPONENTS ONLY; (Indicate Reason and Date Below). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT F:

FURTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date Below) or HIGHER DOD AUTHORITY.

DISTRIBUTION STATEMENT X:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DOD DIRECTIVE 5230.25, WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE, 6 Nov 1984 (Indicate date of determination). CONTROLLING DOD OFFICE IS (Indicate Controlling DoD Office).

The cited documents has been reviewed by competent authority and the following distribution statement is hereby authorized.

A

(Statement)

OFFICE OF NAVAL RESEARCH
CORPORATE PROGRAMS DIVISION
ONR 353
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660

(Controlling DoD Office Name)

(Reason)

(Controlling DoD Office Address, City, State, Zip)

DEBRA T. HUGHES
DEPUTY DIRECTOR
CORPORATE PROGRAMS OFFICE

(Signature & Typed Name)

(Assigning Office)

19 SEP 1995

(Date Statement Assigned)